MASTER'S THESIS

Probabilistic Programming for Spectroscopic Data Analysis

Applied to Vibrational Spectroscopy

van Nispen, J.P.C. (Johan)

Award date:
2020

Probabilistic Programming for Spectroscopic Data Analysis

Applied to Vibrational Spectroscopy

ir. drs. J.P.C. van Nispen

Student:
Date: 23/06/2020

Probabilistic Programming for Spectroscopic Data Analysis

Applied to Vibrational Spectroscopy

by

ir. drs. J.P.C. van Nispen

in partial fulfillment of the requirements for the degree of

Master of Science
in Software Engineering

at the Open University of the Netherlands,
Faculty of Management, Science and Technology
Master's Programme in Software Engineering
to be defended publicly on Tuesday June 23, 2020 at 14:00 PM.

Student number:
Course code: IM9906
Graduation committee: dr. Twan (T. M.) van Laarhoven (Primary supervisor)
dr. Arjen (A. J.) Hommersom (Secondary supervisor)

"Everything should be made as simple as possible, but not simpler."

— Albert Einstein

ACKNOWLEDGEMENTS

Nearing the end of the graduation phase, I would like to take the opportunity to write a few words about my personal experience with the process, and to thank the people who positively contributed to it.

Being a student was not new for me, during the nineties I have already finished two university studies, one in chemistry and one in chemical engineering. However, being a chemist and chemical engineer did not stop me from becoming a professional programmer, and in writing (embedded) software for a living. But, as I discovered over the years, learning to program from a textbook, taking isolated courses, and the practice of *learning-by-doing* gets a person on a very practical level, but for me, I had the feeling that I was still missing something.

About ten years ago, this feeling led me to take up a course in software architecture at the Open University, and it was during this time I first realized that I wanted to give my software related studying efforts more weight and a more solid foundation. For a while this idea remained in the background, but about two years ago I felt it was now or never, and I started studying full-time for the master's degree in software engineering.

The initial plan was to finish the complete program in about one year, but as you can see, it took a little while longer. Being a student in your fifties, is not much like being a student in your twenties, and the road from A to B is never without turns.

This brings me to thanking the people who were close to me during this journey, some of whom often must have wondered as to what I was doing behind my computer screen all this time. My three children, Wilmar, Linde and Marit, thank you for always being joyful, full of life and wonder. My life partner, Annemarie, thank you for always being supportive, constructive and persistent, especially during the periods when I doubted about the road I was on. I thank my father, for initially teaching me the value of knowledge and science. I thank my mother, for demonstrating me the power of a positive attitude and for her lifelong moral support.

Finally, on the professional side, I would like to thank my two supervisors, dr. Twan van Laarhoven and dr. Arjen Hommersom, who both consistently provided me with just the right amount of ideas or information so I could take a new turn when needed, or I could get moving again. Thank you both for providing valuable feedback on the draft versions of this thesis, undoubtedly making it more complete, more logical, and more accessible.

Johan van Nispen
Nijmegen, June 2020

ABSTRACT

Chemometrics is the science of extracting and interpreting chemical data using mathematical and statistical methods. One application of chemometrics is the analysis of spectroscopic data for the classification of chemical mixtures. To improve the performance of the data analysis, a common step is to pre-process the raw data to remove unwanted artefacts originating from instrumental and experimental sources. Data pre-processing usually involves multiple steps, and no clear guidelines exist on how to achieve an optimal result. Moreover, choosing a wrong order of steps can even decrease the data analysis performance.

Results from earlier research show that the use of a simple convolutional neural network is able to surpass the performance of standard chemometric methods on raw vibrational spectral data, and by including a pre-processing step, the analysis performance is increased even more. A drawback of using neural networks for data analysis is the limited model interpretability, while model interpretability is considered an important requirement for application within the chemical domain. Also, spectral datasets often contain a low number of samples, which inherently limits neural network performance.

A potential alternative to the use of neural networks for spectroscopic data analysis, is the use of probabilistic modelling. Probabilistic models are constructed from hidden and observed parameters, which are described by probability distributions. After construction, in a process known as inference, the probabilistic model is conditioned on observed data, and the model prior probability distribution is updated to the model posterior probability distribution. Probabilistic modelling applied within general purpose programming is known as probabilistic programming. The main objective of this research is to explore the usefulness of probabilistic programming for spectroscopic data analysis.

The inference results on spectroscopic datasets show, that with the probabilistic model developed during this research, broad spectral features of a vibrational spectrum can be captured, but that the characteristic spectral features, needed for further data analysis, remain largely unnoticed. Furthermore, the inferred noise level, which represents random noise from the measurement, is found to be much higher than in the observed spectroscopic data, which also decreases the usefulness of the model for spectroscopic data analysis.

To gain insight into the underlying cause of the inferred high noise levels, it was investigated how induced misalignments between the probabilistic model and the data affect the inference outcome. For this purpose, a dataset generator was built with the ability to generate spectral datasets. In a set of scenarios, the effects of induced misalignments between model and data on the parameter inference outcome was systematically investigated. The major effect observed is that, as the misalignment between the probabilistic model and the data grows larger, the inferred noise level also increases.

It is concluded that the current probabilistic model is not yet ready to be used for the data analysis on real-world spectroscopic datasets. A list of recommendations on how to improve the model is provided as future work.

SAMENVATTING

Chemometrie is de wetenschap van het extraheren en interpreteren van chemische data met behulp van wiskundige en statistische methoden. Een toepassing van chemometrie is de analyse van spectroscopische data voor de classificatie van chemische mengsels. Om de doelmatigheid van de data-analyse te verbeteren wordt vaak een voorbewerkingsstap op de onbewerkte data toegepast om zo ongewenste artefacten afkomstig van instrumentele en experimentele bronnen te verwijderen. Deze data voorbewerking omvat vaak meerdere stappen, en er zijn geen duidelijke richtlijnen voor het behalen van een optimaal resultaat. Bovendien kan het kiezen van een verkeerde volgorde de doelmatigheid van de data-analyse verminderen.

Uit eerder onderzoek blijkt dat een eenvoudig convolutioneel neuraal netwerk de doelmatigheid van standaard chemometrische methoden op onbewerkte vibrationele spectroscopische data kan overtreffen. Door het opnemen van een data voorverwerkingsstap wordt deze doelmatigheid nog verder verbeterd. Een nadeel van het gebruik van neurale netwerken voor data-analyse is de beperkte interpreteerbaarheid van het model, terwijl interpreteerbaarheid van het model als een belangrijke voorwaarde voor toepassing binnen het chemische domein wordt gezien. Daarnaast bevatten spectroscopische datasets vaak een beperkt aantal observaties, wat de doelmatigheid van een neuraal netwerkt beperkt.

Een mogelijk alternatief voor het gebruik van neurale netwerken voor spectroscopische data-analyse is het gebruik van probabilistische modellering. Probabilistische modellen zijn opgebouwd uit parameters, die worden beschreven door kansverdelingen. In een proces bekend als inferentie wordt het probabilistische model geconditioneerd op geobserveerde data, en wordt de a-priori kansverdeling van het model bijgewerkt naar de a-posteriori kansverdeling. Probabilistische modellering toegepast binnen programmeren staat bekend als probabilistisch programmeren. Het doel van dit onderzoek is om de bruikbaarheid van probabilistisch programmeren voor spectroscopische data-analyse te verkennen.

De resultaten op spectroscopische datasets laten zien dat het ontwikkelde probabilistische model, globale kenmerken van een vibrationeel spectrum kan modelleren, maar dat de karakteristieke kenmerken, noodzakelijk voor verdere data-analyse, worden gemist. Bovendien blijkt het gevonden ruisniveau, dat de uit de meting afkomstige willekeurige ruis voorstelt, veel hoger dan in de geobserveerde spectroscopische data, waardoor toepassing van het model voor spectroscopische data-analyse eveneens wordt bemoeilijkt.

Om meer inzicht te verkrijgen in de oorzaak van het hoge ruisniveau, werd onderzocht hoe afwijkingen tussen het probabilistische model en de data de uitkomst van inferentie beïnvloed. Hiervoor is een data generator gebouwd met de mogelijkheid om spectroscopische datasets te genereren. In verschillende scenario's werd systematisch het effect van geïnduceerde afwijkingen tussen het model en de data op de uitkomst van de inferentie onderzocht. De belangrijkste observatie is dat naarmate de afwijking tussen het probabilistische model en de data groeit, ook het ruisniveau toeneemt.

De conclusie van het onderzoek is dat het ontwikkelde probabilistische model nog niet geschikt is om te worden toegepast voor de analyse van spectroscopische datasets. Ter verbetering van het model is een lijst met aanbevelingen voor toekomstig werk opgenomen.

CONTENTS

List of Figures

List of Tables

1

INTRODUCTION

1.1. SPECTROSCOPIC DATA ANALYSIS

INFRARED SPECTROSCOPY

Infrared (IR) spectroscopy is the study of the interaction of infrared light and matter. Certain frequencies of infrared light cause the *vibrational motion* of the atoms of a molecule, like stretching, bending and twisting, to excite (Stuart, 2004). When light of these frequencies is passed through a chemical sample, the light is *absorbed*. As the absorption frequency of infrared light is specific for a chemical bond or functional group in the molecule, IR spectroscopy is often used to determine the presence or concentration of specific molecules in a chemical sample, or to *fingerprint* chemical mixtures.

An IR spectrum is a graph which plots the frequency of infrared light against the absorbance or transmittance through a chemical sample. An example of this is shown in Figure 1.1.

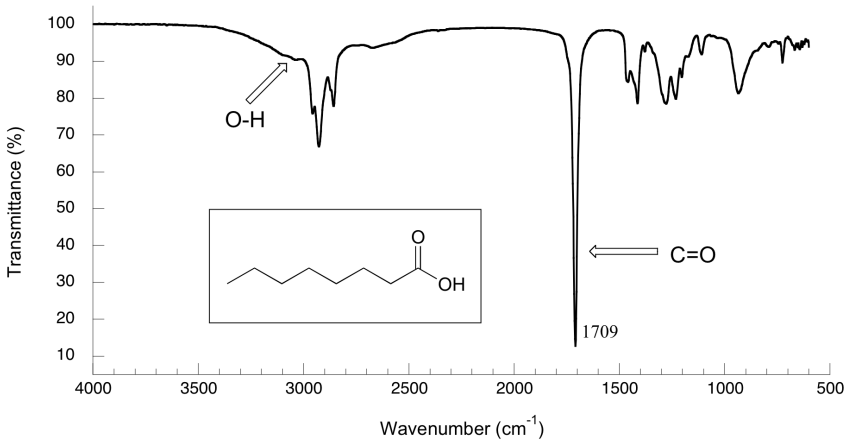


Figure 1.1: IR spectrum of octanoic acid ($C_8H_{16}O_2$).

CHEMOMETRICS

Chemometrics, or chemometric analysis, is the science of extracting and interpreting information from data produced by chemical analytical instruments, by using mathematical

and statistical techniques (Massart et al., 1998). A widely used application of chemometrics is the analysis of spectroscopic data for the *classification* of chemical mixtures. The task here is to map the spectra of these mixtures (input objects) into class assignments (output objects), e.g. sample X belongs to class A or B. Example applications of this task are found in a wide range of industries, including agriculture, pharmaceutical, petrochemical, food science and medicine. Some popular chemometric methods used in qualitative and quantitative spectral analysis are Principal Component Analysis (PCA), Partial Least Squares Regression (PLS), Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM) (Engel et al., 2013).

DATA PRE-PROCESSING

To improve the performance of the data analysis, raw spectral data is typically pre-processed into cleaned data to remove unwanted artefacts originating from instrumental and experimental sources. Data pre-processing usually involves a number of steps, including baseline correction, scatter correction, noise removal and scaling (Engel et al., 2013; Rinnan, Van Den Berg, & Engelsen, 2009). The order in which the pre-processing steps are applied depends on the analysis goal, the spectroscopic acquisition technique and dataset, and has a profound influence on the performance of the subsequent multivariate analysis. An example of the artifacts which are typically found in IR spectra is shown in Figure 1.2.

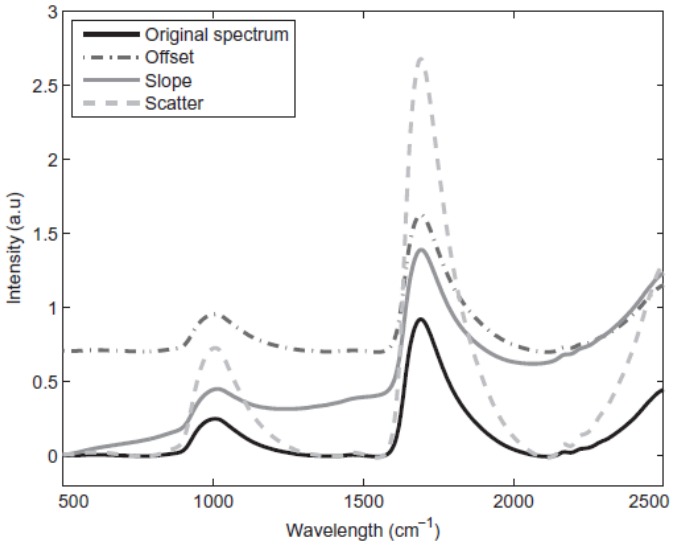


Figure 1.2: Examples of different artifacts in spectroscopic data (Engel et al., 2013).

Although choosing the right order of pre-processing steps is important, there are no clear guidelines as when to use which steps and moreover, choosing an inappropriate order of steps might even lead to a decrease in analysis performance (Engel et al., 2013). To reduce the need for a hand-crafted, and sometimes complex data pre-processing strategy, there is an ongoing search within the chemometric community for techniques which are able to give good results on raw, often noisy and/or highly correlated spectral data.

CONVOLUTIONAL NEURAL NETWORKS

In the past few decades there has been a significant effort in exploring the use of artificial neural networks (ANN) for spectroscopic data analysis (Marini, 2009). However, due to

difficulties with overfitting and the inability to interpret the underlying model (the model was considered a black box), the interest in this particular type of neural network has faded. More recently, advances in machine learning have demonstrated that another type of ANN architecture, called a Convolutional Neural Network (CNN), is able to learn interpretable representations from image, video, speech and audio data (LeCun, Bengio, & Hinton, 2015).

When a CNN is applied to spectroscopic data, it was demonstrated that a very simple CNN architecture, using only a single convolutional layer, is able to outperform the classification accuracy of standard chemometric analysis methods such as Partial Least Squares Regression Linear Discriminant Analysis (PLS-LDA), k-Nearest Neighbors (kNN) and Logistic Regression (LogReg), both when used on pre-processed and on non pre-processed data (Acquarelli et al., 2017). This study also showed that, although the classification performance results on non pre-processed data has improved as compared to standard methods (86% by CNN as compared to 62% by PLS), an even better performance could be achieved when using an optimal data pre-processing strategy (96% by CNN as compared to 89% by PLS).

The spectroscopic datasets used in this study are relatively small (compared to datasets typically used in deep learning), often containing only a few tens of samples per dataset. This probably limits the performance of the neural network used. The amount of spectroscopic applications and portable spectroscopic devices is increasing, and it is expected that the rapidly growing volume of spectral data (i.e. increased dataset size) will benefit the performance of CNN based chemometric analytical methods. A review of the current progress of CNN based analytical methods, and a practical guide on developing neural network based methods for spectroscopic analysis has recently been provided by (Yang et al., 2019).

Next to the small spectral dataset size, a major limitation in using neural network based models for spectroscopic analysis is poor model interpretability. Unknown environmental and instrumental influences induce spectral artefacts, which cause variable selection techniques to contain uninformative variables, lowering the models prediction performance (Yun, Li, Deng, & Cao, 2019). The ability to identify important spectral regions, linked to the presence or concentration of the active chemical substances in the mixture, is considered an essential ingredient for a reliable and consistent model (Acquarelli, van Laarhoven, Jansen, Buydens, & Marchiori, 2019). Together with improving neural network learning repeatability, improved model interpretability is seen as one of two factors in advancing further promotion of spectroscopic analysis (Yang et al., 2019). The original ANN based models were considered as a black box, and therefore lacked any model interpretability. In the study by Aquarelli et al., stability selection was used as a feature selection technique to identify important spectral regions in the input spectrum (Acquarelli et al., 2017; Meinshausen & Bühlmann, 2010). The study demonstrates that, when the model has a high performance, interpretation is possible for a CNN based architecture, very often finding the correct important regions which are in line with prior knowledge. However, in a later assessment, the authors note that the specific stability selection technique might be prone to missing important regions under certain circumstances (Acquarelli et al., 2019).

BAYESIAN MODELLING

An alternative approach to the use of a CNN for spectroscopic data analysis is a technique called Bayesian modelling, or probabilistic modelling. A potential advantage of this technique is that it has better model interpretability, and is known to work well on datasets

where only a small number of samples are available. In the Bayesian modelling approach, a model is constructed from unobserved and observed parameters which are connected through a directed acyclic graph (DAG), forming a probabilistic graphical model known as a Bayesian network.

Central to probabilistic modelling is the use of probability theory to describe the model parameters. The exact value of a model parameter is therefore inherently uncertain, and is described by a probability distribution. After model construction, the observed data and the model are combined in a process called conditioning, to which Bayes' theorem is central. The conditioning process transforms the models initial prior probability distribution into an updated distribution conditioned on the observed data, the posterior probability distribution. Once conditioned on the observed data, the model can be used to make predictions about future data observations.

Probabilistic machine learning and Bayesian modelling have gained an increasing momentum during the last couple of years (Ghahramani, 2015). The rising application of probabilistic modelling would not have been possible without major advances in numerical computing and the development of Probabilistic Programming Languages (PPL) (van de Meent et al., 2018), which allow for the expression of probabilistic models as computer programs and automatic Bayesian inference of these models.

1.2. RESEARCH OBJECTIVE

The previous section gave an overview and showed the current limitations of CNN based spectroscopic data analysis. To summarize, the most important limitations are:

1. The use of a simple CNN is able to outperform standard chemometric analysis methods on raw spectral data. However, the addition of an optimal data pre-processing step still improves the analysis result.

2. The number of available samples in spectroscopic datasets is often very small, as compared to the size of the datasets used in neural network design for image, video, speech and audio applications, which limits the CNN based prediction performance.

3. The chemometric community considers model interpretability an important feature. This is often non-optimal or absent when using neural networks for data analysis.

To overcome these limitations, one possibility is to explore the use of Bayesian modelling and probabilistic programming in the analysis of raw spectral data. The overall research objective of the current research is therefore formulated as follows:

Can probabilistic programming be used for spectroscopic data analysis?

In order to help answer the research objective, the following research questions have been formulated:

RQ1: *Can a probabilistic model be constructed that captures the characteristic features of a vibrational spectrum?*

This question aims at designing and validating a probabilistic model that is able to capture the most characteristic features of an IR spectrum. Traditionally, IR spectroscopy is most often used to establish the presence or concentration of a single

chemical substance. Distinct chemical bonds and functional groups show their presence by displaying specific peaks, or a pattern of peaks in the IR spectrum. Also, between samples of the same dataset, the acquired spectra can show a large variation in the background signal. The Bayesian model must therefore take into consideration all these underlying, hidden factors.

RQ2: *What is the effect of a misalignment between the probabilistic model and the spectral data on the inference outcome?*

A central part of Bayesian analysis is about building and evaluating models. A model describing reality is always an abstraction which tries to capture only the most relevant part of a data generating process. So, inherently there will always be a difference between the model and the data. Therefore, an important question is to explore what potential effects differences between the probabilistic model and the observed data have on the inference outcome.

1.3. RESEARCH METHOD

The research started by conducting a literature search for relevant publications concerning the probabilistic modelling of vibrational spectra. The results of this search gave insight into what relevant research has already been done on the subject, and also into what kind of model structure, parameters and priors would be useful in building probabilistic models for spectral analysis.

In the next stage, the insights provided by the literature search were used to build a probabilistic model for spectral analysis. However, results of experiments on real-world datasets revealed that the inference process often did not converge, and that model performance was often also quite poor. It was decided to focus on creating controlled differences between the probabilistic model and the data, which allows for a more systematic study of the effects of model-data differences on the inference outcome.

In the final stage, a dataset simulator was built which had the ability to generate vibrational spectra. Also, a set of experimental scenarios was constructed which are based on controlled misalignments between model and data. The scenarios all share the fact that only a single parameter value in the data generating process is varied per scenario, while the rest is kept constant. Proceeding in this way makes it possible to observe and measure the effect on the inference outcome of induced differences between model and data.

1.4. THESIS OUTLINE

The research described in this thesis is organised as follows: Chapter 2 provides background information on Bayesian modelling, probabilistic programming, spectral line shapes and model evaluation. Chapter 3 presents an overview of related work concerning Bayesian modelling of spectroscopic data that can be found in recent literature. In Chapter 4 the design of the dataset simulator, probabilistic model and experimental scenarios is described, and Chapter 5 presents the results of the experimental scenarios. The thesis ends with Chapter 6, where the conclusions are presented and recommendations for future work are given.

2

BACKGROUND

In order to better understand the modelling approach presented in this research, this chapter provides background material on Bayesian modelling, probabilistic programming, spectral line shapes and model evaluation and convergence. The most relevant aspects of Bayesian modelling and analysis, i.e. Bayes' theorem, probability distributions, joint probability distribution, probabilistic graphical models (PGM), Bayesian networks and Kruschke diagrams will be introduced first. Next, probabilistic programming and (automatic) inference engines will be described. The chapter ends by introducing a probabilistic programming environment, PyMC3 (a library that adds probabilistic programming capabilities to Python), by briefly introducing the physical principles behind IR and Raman spectral line shapes, and by providing a short overview of common tools used in model evaluation and convergence.

Further background and reference material on Bayesian modelling and probabilistic programming can be found in the books: Bayesian Analysis with Python (Martin, 2018), Bayesian Methods for Hackers (Davidson-Pilon, 2015), Pattern Recognition and Machine Learning (Chapter 8) (Bishop, 2006) and Practical Probabilistic Programming (Pfeffer, 2016).

2.1. BAYESIAN MODELLING

Bayesian modelling, or probabilistic modelling, is a model-based machine learning technique. In general, modelling is about creating a simplified description of a system or process, thereby focusing only on the most relevant parts to explain the properties of interest. Once created, the models are used to make predictions, which can be compared to real-world observations. A model is said to be well defined if the model predictions are in line with the observations. Usually there is *uncertainty* at many levels in modelling: about the structure of the model, about what values the parameters in the model should have, and also in the observed data, which most often contains noise.

In Bayesian modelling uncertainty is used as a structural building block, and the mathematics of probability theory are used to represent and manipulate the uncertainty. Bayesian models are composed by using model parameters which can have a range of values, each value having a separate probability. Mathematically this range of values is represented as a *probability distribution*. When the model parameters (vertices) and the relationships between them (edges) are represented as a DAG these models form a PGM called a Bayesian network (see Section 2.1.2).

Central to Bayesian modelling is the use of Bayes' theorem to update the model parameters in a process called conditioning. Conditioning allows for the *inference* of the unobserved model parameter values, given the observed data. Bayes' theorem is discussed in Section 2.1.1. For all but the most simple models, inference of model parameters cannot be done analytically, but is done by numerically approximating the model parameter values by running simulations using inference algorithms. (Automatic) inference engines are discussed further in Section 2.2.1. The conditioning process transforms the models parameters *prior probability distributions* (before seeing the data) into the *posterior distributions* (after seeing the data). Conditioning is the way by which a Bayesian model "learns" from new data.

BOX'S LOOP

Probabilistic modelling is often compared to cycling through Box's loop, which focussed on the iterative aspects of the scientific method (Box, 1976). An adaption of this loop applied to probabilistic modelling for solving data analysis problems, is shown in Figure 2.1.

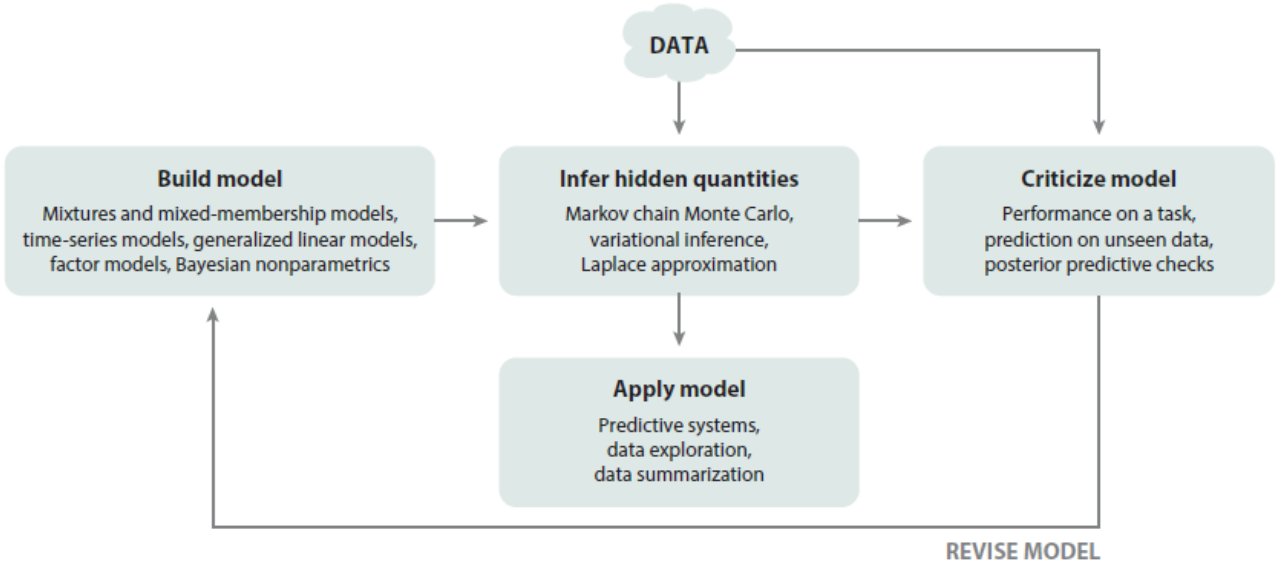


Figure 2.1: Box's loop (Blei, 2014)

The cycle starts by building a probabilistic model which could explain the observed phenomena. Second, given the dataset and model, an inference algorithm is used to approximate the posterior distribution. In the third step, the posterior distribution is used to test and criticize the model against the data. If the model produces results which are in line with the observations, the model can be used and applied, in case the results are seen as unsatisfactory, the model is revised and the cycle repeats from step one.

2.1.1. BAYES' THEOREM

Bayes' theorem is derived from the definition of conditional probability. Consider the following probability space S, with the probability function $P(\cdot)$ assigning a probability to each

subset of S such that:

$$\begin{aligned} &\text{I. } P(A) \geq 0 && \text{for each } A \subset S, \\ &\text{II. } P(A \cup B) = P(A) + P(B) && \text{if } A \cap B = \emptyset, \\ &\text{III. } P(S) = 1. \end{aligned}$$

In this space S, given events A and B, the *conditional probability of event A, given event B* is defined as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A,B)}{P(B)} \tag{2.1}$$

Where $P(A,B)$ is the *joint probability* of events A and B. Rearranging (2.1) leads to the *product rule* for probabilities:

$$P(A,B) = P(A|B) \cdot P(B) \tag{2.2}$$

As $P(A,B) = P(B,A)$, (2.2) can also be written as:

$$P(B,A) = P(B|A) \cdot P(A) \tag{2.3}$$

Combining (2.2) and (2.3) leads to the definition of Bayes' theorem:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.4}$$

Equation (2.4) allows for the calculation of the conditional probability $P(A|B)$ from the opposite conditional probability $P(B|A)$ and the probabilities $P(A)$ and $P(B)$, which is often more convenient to work with.

When working with conditional probabilities it is important to realise that $P(A|B)$ (the probability of event A, given B) is usually not the same as $P(B|A)$ (the probability of event B, given A), but can be greater, equal or smaller. This is illustrated by a thinking of a very simple example (Martin, 2018): The probability of a person being the Pope, given that he is Argentinian, i.e. $P(Pope|Argentinian) \simeq 1/44.000.000$, is not he same as the probability of a person being Argentinian, given he is the Pope, i.e. $P(Argentinian|Pope) = 1$.

The conditional probabilities in (2.4) equally apply to any probability distribution over the events A and B. In the case of a Bayesian model, event A represents a *hypothesis* and event B represents *evidence* (observed data). The hypothesis in a Bayesian network is the joint distribution over the model parameters (see Section 2.1.2). In the context of Bayesian modelling, the goal is to update the model parameter distributions, in light of the observed data. Written in these terms, (2.4) can be equivalently formulated as follows:

$$P(\theta|y) = \frac{P(y|\theta) \cdot P(\theta)}{P(y)} = \frac{P(y,\theta)}{\int_\theta P(y) \cdot d\theta} \tag{2.5}$$

Where θ is the hypothesis, and y is the observed data. The marginal probability of θ, $P(\theta)$, is the probability distribution before seeing the data, and is called the **prior distribution**. The conditional probability of the observations y, given the parameter θ, $P(y|\theta)$, is called the **likelihood**, and defines how observations are determined by the parameters. The conditional probability of θ, given the observations y, $P(\theta|y)$, is called the **posterior distribution**, and reflects all model knowledge after seeing the data. The **marginal likelihood**, or evidence, $P(y)$, is the probability of seeing the data, averaged over all possible

values of θ. Since $P(y)$ is a normalizing constant, it can be factored out of (2.5), after which we get the following proportionality:

$$P(\theta|y) \propto P(y|\theta) \cdot P(\theta) \tag{2.6}$$

This equation is read as: *posterior probability* $\propto$ *likelihood* $\cdot$ *prior probability*. Equation (2.6) reflects the central use of Bayes' theorem in Bayesian modelling, which is the posterior distribution being the updated prior distribution, after being exposed to (new) data.

2.1.2. BAYESIAN NETWORKS

A model in the Bayesian modelling context is a specification of the joint probability distribution over all parameters in the model. For example, consider the joint distribution $P(a,b,c)$ over the three variables a, b, and c. By applying the product rule (2.2) this can be written as:

$$P(a,b,c) = P(c|a,b) \cdot P(a,b) \tag{2.7}$$

Applying the product rule to (2.7) a second time yields:

$$P(a,b,c) = P(c|a,b) \cdot P(b|a) \cdot P(a) \tag{2.8}$$

As read from (2.8), the joint probability distribution $P(a,b,c)$ can be written as the product of conditional probabilities $P(x|y)$ and a marginal probability $P(z)$. This result is general is applicable to discrete and continuous variables, and applies to all functional forms of probability distributions over the model variables, e.g. Gaussian, Bernoulli, beta, gamma, etc.

The joint distribution $P(a,b,c)$ can be visualized by constructing a DAG from the terms on the right-hand side of (2.8). In this graph, each node represents a variable from the joint distribution, and for each conditional relationship between the variables a directed arrow is added between the nodes. In the relationship $a \rightarrow c$, node a is called the *parent* of c, and node c is called the *child*. For factor $P(c|a,b)$, one arrow from a and one from b are drawn, $P(b|a)$ has one arrow from a, and $P(a)$ has no incoming links. The result is shown in Figure 2.2.

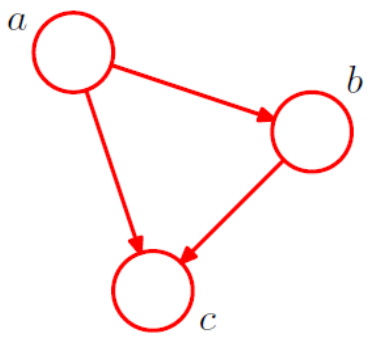


Figure 2.2: Fully connected directed acyclic graph over three variables (Bishop, 2006).

The result of the current example can be applied to the joint distribution over K variables, given by $P(x_1, x_2, \ldots, x_K)$ by repeated application of the product rule. The result is a product of conditional probability distributions, with one for each variable. When represented graphically, this results in a *fully connected* graph, with a link between each pair of nodes. The result is valid for any choice of distributions.

It is however the *absence* of links in the graph that holds information about the distribution properties that the graph represents. As an example, the graph in Figure 2.3 shows a non-fully connected graph over seven variables. It is not fully connected because there are no links between x_1 and x_2, or from x_3 to x_7, for example.

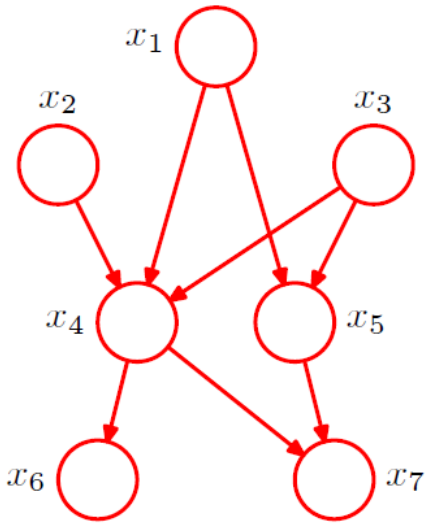


Figure 2.3: Non-fully connected directed acyclic graph over seven variables (Bishop, 2006).

The graph represents the joint distribution over all variables, as a product of conditional distributions, where each node is conditioned only by its parent nodes. The full joint distribution, $P(x_1, x_2, \ldots, x_7)$, is given by:

$$P(x_1) \cdot P(x_2) \cdot P(x_3) \cdot P(x_4|x_1, x_2, x_3) \cdot P(x_5|x_1, x_3) \cdot P(x_6|x_4) \cdot P(x_7|x_4, x_5) \tag{2.9}$$

The factorization of the joint probability in (2.9) can be extended to K nodes. In this case, the resulting distribution for a Bayesian network with K nodes can be written as:

$$P(x_1, x_2, \ldots, x_K) = \prod_{k=1}^{K} P(x_k|Predecessors(x_k)) \tag{2.10}$$

Where $Predecessors(x_k)$ is a function returning the set of parent nodes of x_k. The key concept expressed by (2.10) is that the full joint probability distribution over a (potentially very large) set of variables can be calculated as a product of factors, with each factor normally only depending on a few number of variables. This factorisation results in a considerable simplification of the problem, and makes probabilistic models computationally manageable.

The nodes in Figures 2.2 and 2.3 show the unknown, or *hidden* variables of the model, and are represented as open circles. In contrast the known, or *observed* variables of the

model are usually drawn as shaded circles. When multiple observations are made, the graphical representation is made more compact by drawing a *plate* around the variable, showing only one observation from the set and labeling the plate with N. Also, when a parameter is influenced by a fixed variable, called a *hyperparameter*, this parameter is represented as a small solid circle. An example of this graphical representation is shown in Figure 2.4.

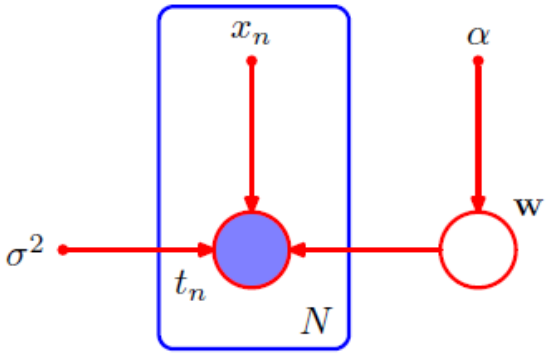


Figure 2.4: Example of a PGM including plate, hyperparameters, data and hidden variables (Bishop, 2006).

2.1.3. KRUSCHKE DIAGRAMS

When analysing and reporting probabilistic models it is often insightful to also show the probability distributions that the model is using for the random variables it is composed of. While the graphical representation of Bayesian networks shown in Section 2.1.2 captures the relationships and dependencies of between the different variables, it does not display the specific probability distribution chosen for each node.

A diagram style which explicitly displays the probability distributions as part of the model is the Kruschke diagram (Kruschke, 2014), of which a short example is given next. Consider the statistical equations below, which describe a simple beta-binomial, or coin-flipping, model (Martin, 2018):

$$\begin{aligned} \theta &\sim \text{Beta}(\alpha, \beta) \\ y &\sim \text{Binom}(n = 1, p = \theta) \end{aligned} \tag{2.11}$$

Reading from equation (2.11), the observed variable y (0 or 1, head or tails) is distributed as a binomial distribution, with parameters $n = 1$ and $p = \theta$ (the coin fairness), and θ is distributed as a beta distribution, characterized by the parameters α and β. When (2.11) is represented as a Kruschke diagram, Figure 2.5 appears. Drawing probabilistic models in this style more directly displays the relationships and probability distribution specific information about the model.

2.2. PROBABILISTIC PROGRAMMING

To a large variety of problems the application of Bayesian modelling and analysis is very simple in concept. On the one hand there are the *known* parameters, i.e. the observations y, or data, and on the other hand there are the *unknown* parameters, to which a prior

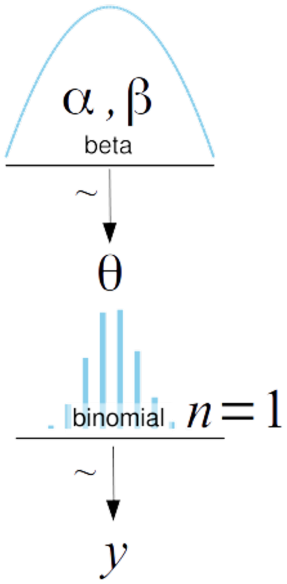


Figure 2.5: Kruschke diagram for the beta-binomial model (Martin, 2018).

probability distribution $P(\theta)$ is assigned. Building a probabilistic model involves making assumptions on how the parameters influence and relate to each other. As was shown in Section 2.1.2, the probabilistic model can be graphically visualised as a Bayesian network, which represents the full joint probability distribution $P(y,\theta)$. Bayes' theorem (see Section 2.1.1) is then used to transform the prior probability distribution $P(\theta)$, into the posterior distribution $P(\theta|y)$, normally reducing the uncertainty in the unknown parameters. The difficulty lies in the fact that for all but the most simple probabilistic models this approach leads to analytically unsolvable expressions.

It was only recently that advances in numerical methods have led to the development of universal inference engines that can, in principle, solve any probabilistic model. This has led to the development of **probabilistic programming languages (PPL)**. The main advantage of using a PPL is that the process of creating and solving a probabilistic model has been completely separated, which is less time-consuming, less error-prone, and promotes model-based thinking. Probabilistic models are created in a few lines code, and are solved by automatic Bayesian inference. It is believed that probabilistic programming will have a major impact on machine learning and scientific modelling (Ghahramani, 2015).

INTUITIVE VIEW

The process of probabilistic programming can be intuitively visualized as is depicted in Figure 2.6. The left hand side of the figure shows the traditional computer science (CS) programming process. In this approach a program is written first, and the value of the parameters is fixed. Next, the program is evaluated and some output is produced. On the right hand side of Figure 2.6, the traditional statistical modelling approach is shown. This process starts with the observations y and the creation of a probabilistic model $P(y,\theta)$, which can account for the observations. Next, using algebra and inference techniques, the posterior distribution $P(\theta|y)$ is characterized. In the middle of Figure 2.6 the probabilistic programming program flow is shown. In this approach, using traditional computer science

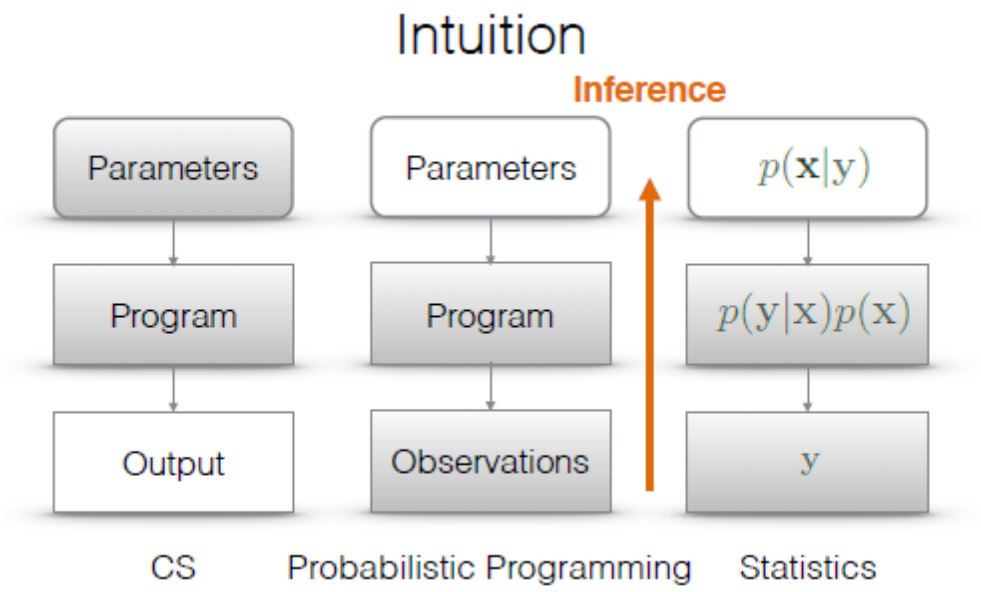


Figure 2.6: Intuitive view of probabilistic programming (van de Meent et al., 2018).

tools, a probabilistic model which explains the observations is first programmed in a probabilistic programming language, after which the model is solved using automatic statistical inference engines. The outcome of this programming flow is the posterior distribution of the program input parameters, which could have produced the observed program output.

2.2.1. (AUTOMATIC) INFERENCE ENGINES

As noted earlier, for all but the simplest Bayesian models, computing the posterior distribution, $P(\theta|y)$ (see Eq. 2.5), over the parameters in the model is practically impossible. The main reason for this is that the marginal likelihood, $P(y)$, often involves solving a very computationally expensive integral. To overcome this limitation the posterior distribution can be approximated using various numerical techniques. Recent advances in this field have led to classes of algorithms that can, in principle, be used to approximate the posterior distribution of any probabilistic model. This has led to the development of probabilistic programming languages and **(automatic) inference engines**, which allow for a complete separation between the tasks of model-building and model-solving.

In general, the algorithms used in the inference engines can be divided in two classes, *Markov chain Monte Carlo (MCMC)* methods, which are based on sampling, or *Variational Inference (VI)* methods, which are based on optimization. Both classes will briefly be discussed next.

MARKOV CHAIN MONTE CARLO

The MCMC method is based on the principle of *sampling*. The algorithm works by sampling values for the parameters θ, calculating the value for the likelihood $P(y|\theta)$ and prior $P(\theta)$, and subsequently calculating the posterior $P(\theta|y)$ value. As values are sampled according to their probability, more samples are drawn from high-probability regions then from low-probability regions. This principle has been visualised in Figure 2.7 for two continuous parameters x and y, with values between 0 and 1.

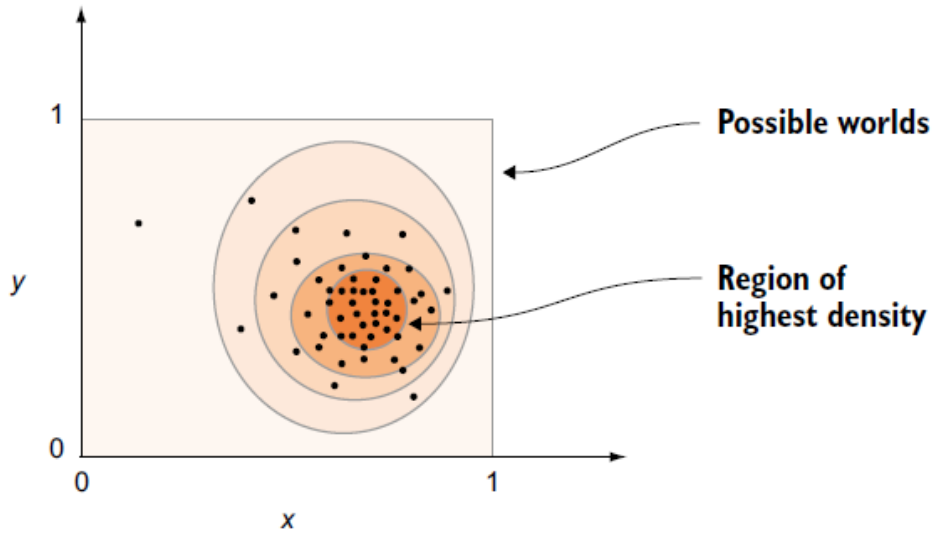


Figure 2.7: Covering the space of possible worlds using sampling (Pfeffer, 2016).

In Figure 2.7, regions of high-probability density are shown in darker color then regions with low-probability density. The regions are covered with a set of samples, and the density of the samples is an approximation of the probability density in this region. With an infinite amount of samples, the sampled posterior distribution will become equal to the true posterior distribution.

The principle of approximating the true probability density by sampling, can also be formulated mathematically. The statistical expected value is given as:

$$\mathbb{E}|f| = \int_{\theta} P(\theta) \cdot f(\theta) \cdot d(\theta) \tag{2.12}$$

By using MCMC sampling, equation (2.12) can approximated by taking a finite amount of samples, which is written as:

$$\lim_{N \to \infty} \mathbb{E}|f| = \frac{1}{N} \cdot \sum_{n=1}^{N} f(\theta_n) \tag{2.13}$$

The main challenge in applying the MCMC method is to *converge* to the true expected value with the least amount of samples, or *draws*.

The MCMC method name is composed from two parts: *Monte Carlo* and *Markov chain*. The first part of the name, Monte Carlo, originates from the fact that the sampling process takes draws at random from probability distributions. The second part, Markov chain, originates from the fact that sampling is done from a object known as a Markov chain.

A Markov chain is a sequence of states and a set of transition probabilities which describe how to move between the states. Furthermore, the probability of moving from one state to any other state depends only on the current state. By picking a starting point and moving to the next state according to the transition probabilities a *random walk* is made over the combined distribution. By further assuring that the Markov chain complies to a special condition called the **detailed balance condition**, or reversibility condition, which says that the probability of moving from state i to j, is equal to the probability of moving

from state j to i, it can be proven that sampling from the chain will approximate sampling from the true posterior distribution.

A few of the most widely used sampling algorithms, which are all based on the MCMC method, are briefly discussed below:

- **Metropolis-Hastings (MH)**: Conceptually the MH algorithm (Hastings, 1970) works by starting out from an initial parameter value, and choosing a new parameter value according to a *proposal distribution*, which is easy to sample from. In the next step the new parameter value can be *accepted* or *rejected*, according to the *Metropolis-Hastings criteria*, which says that the new parameter value is accepted according to the ratio of the probabilities of the old and new parameter values. This procedure assures that high probability values are often accepted, which leads to a more efficient sampling process. Furthermore, the output of the process is a list of sampled parameter values, which approximate the posterior distribution when given enough samples.

- **Hamiltonian Monte Carlo (HMC)**: A major drawback of the MH algorithm is the fact that it can take a long time (i.e. it can take a lot of samples) before the algorithm approximates the posterior distribution. The HMC algorithm (Duane, Kennedy, Pendleton, & Roweth, 1987) improves the MH algorithm by not picking the new proposed parameter values at random, but by following the *curvature* of the parameter space. This process leads to a more efficient sampling method as compared to the MH algorithm, because the probability of accepting the new proposed parameter value is higher. As the HMC algorithm needs to compute the gradient of the parameter space at each step, it is computationally more expensive than the MH algorithm. In order for HMC to work efficiently, the algorithm needs to be tuned by setting a step size ϵ and a desired number of steps L, which requires an experienced used. Recently, the **No-U-Turn Sampler (NUTS)** method has been proposed, which finds the optimal values for ϵ and L by itself, eliminating the need for hand-tuning, and thus making the method suitable as an automatic inference engine (Hoffman & Gelman, 2014). Because the NUTS method uses gradients, it only works for models composed of continuous distributions.

- **Sequential Monte Carlo (SMC)**: Both the HM and HMC methods are not well suited for posterior distributions which have multiple peaks separated by regions of low probability. To overcome this limitation, the SMC algorithm (Doucet, De Freitas, & Gordon, 2001) uses the idea of *tempering*. Using this idea, (2.6) is rewritten as:

$$P(\theta|y)_\beta \propto P(y|\theta)^\beta \cdot P(\theta) \tag{2.14}$$

  Where β is the tempering parameter. When $\beta = 0$, the tempered posterior $P(\theta|y)_\beta$ is equal to the prior $P(\theta)$, and when $\beta = 1$, the tempered posterior is equal to the full posterior distribution. In general, it is easier to sample from the prior than from the posterior, so by starting from $\beta = 0$ and slowly increasing it, the distribution is *morphed* from easy to complex. The SMC algorithm starts by generating a set of samples from the tempered posterior, increasing β next, and computing a set of weights based on the new posterior distribution. The samples are resampled according to

their weight, which removes samples with a low probability and replaces them with samples with a high probability. Next, to explore the parameter space, a Metropolis step is taken from each new sample, and the algorithm repeats with increasing β until $\beta \geqslant 1$.

VARIATIONAL INFERENCE

The basic idea behind variational inference is to approximate the posterior probability distribution by a simpler distribution (Blei, Kucukelbir, & McAuliffe, 2017). This approach tends to be faster for large datasets or very complex models.

The difference, or *closeness*, of the approximate posterior distribution to the posterior distribution is measured as the Kullback-Leibler (KL) divergence, which is expressed as:

$$D_{KL}(q(\theta)||P(\theta|y)) = \int q(\theta) \cdot log\frac{q(\theta)}{P(\theta|y)} \cdot d(\theta) \tag{2.15}$$

In this equation, $q(\cdot)$ is the simpler distribution used to approximate the posterior distribution $P(\theta|y)$. By optimizing the parameters of $q(\cdot)$, the KL divergence is minimized.

Since the posterior distribution $P(\theta|y)$ is unknown, (2.15) cannot be used directly. Rewriting creates an alternative formulation:

$$D_{KL}(q(\theta)||P(\theta|y)) = \underbrace{-\int q(\theta) \cdot log\frac{P(\theta,y)}{q(\theta)} \cdot d(\theta)}_{evidence\ lower\ bound\ (ELBO)} + log(P(y)) \tag{2.16}$$

As $D_{KL} \geqslant 0$, maximizing the ELBO in (2.16) leads to a minimization of the KL divergence.

The distribution $q(\cdot)$ to approximate the posterior $P(\theta|y)$ can be any probability distribution. One solution is to approximate the high-dimensional posterior $P(\theta|y)$ by a product of independent, one dimensional probability distributions. This can mathematically be expressed as:

$$q(\theta) = \prod_j q_j(\theta_j) \tag{2.17}$$

This approach is known as the *mean-field approximation*. In theory, a different distribution could be chosen for each $q_j(\theta_j)$. In practice however, most often members of the exponential family of distributions are chosen, e.g. normal, exponential, beta, gamma, Poisson, Bernoulli, Dirichlet, etc. The formulation of the inference problem in terms of a product of simple distributions, and maximization of the ELBO has turned it into a *optimization* problem.

However, the described approach is a recipe for solving a single model only. In practice models are often changed in an iterative manner, and solving each model manually at each step is too slow. To turn the approach into an automatic and universal inference engine, which can be used on a broad class of models, the method of **Automatic Differentiation Variational Inference (ADVI)** (Kucukelbir, Tran, Ranganath, Gelman, & Blei, 2017) has recently been proposed.

2.2.2. PyMC3

An increasing number of probabilistic programming languages have appeared over the last few years, of which a short overview and discussion can be found in (van de Meent et al.,

2018; Ghahramani, 2015). Here, we will only focus on one of the most widely used probabilistic programming languages, PyMC3, as this language has been chosen as the implementation language for the current research.

PyMC3[1] is a open source Python library for probabilistic programming (Salvatier, Wiecki, & Fonnesbeck, 2016). It features an expressive, clean and intuitive syntax, which comes close to the model denotation used in statistical literature to specify probabilistic models. The code base for PyMC3 is written in Python, but for computationally demanding tasks it uses Theano[2] to compute gradients via automatic differentiation, and to compile Python code to C, which boosts execution speed. (Automatic) Bayesian inference can be performed by either MCMC based methods (e.g. NUTS, MH or SMC) or VI based methods (e.g. ADVI). An example of how a probabilistic model is coded in the PyMC3 environment, consider the coin-flipping model from (2.11). The code listing is shown in Listing 1.

```
import pymc3 as pm
# observations
data = [1, 0, 0, 0]
with pm.Model() as model:
    # prior distribution
    theta = pm.Beta('theta', alpha=1., beta=1.)
    # likelihood of observations
    y = pm.Bernoulli('y', p=theta, observed=data)
    # sample from the posterior distribution
    trace = pm.sample(1000)
```

Listing 1: Code listing for the beta-binomial model (Martin, 2018).

The statistical model of (2.11) can very intuitively be translated into Python code. After specifying the prior distribution `theta` and the likelihood `y`, which is conditioned on the observations by setting the argument `observed=data`, inference can be run by asking for a trace of 1000 samples from the posterior distribution. The samples are stored in the `trace` object.

2.3. SPECTRAL LINE SHAPES

As mentioned in the introduction, when molecules absorb infrared light of a specific energy, the molecules get into an excited state. When falling back from the exited state into the ground state, the absorbed energy is emitted again. As the molecules are surrounded by a large number of neighbouring molecules, the slightly varying interactions between them gives rise to a distribution of vibrational frequencies. The observed *vibrational* (IR and Raman) spectral line shape is the sum of these collective interactions of molecules absorbing and scattering light (Bradley, 2007).

The functional form of the peak can be represented as a continuous function, which is shaped by underlying physical properties of vibrational spectroscopy. In IR and Raman

[1] https://docs.pymc.io/
[2] http://deeplearning.net/software/theano/

spectroscopy, two physical effects predominantly contribute to the broadening of spectral lines, *Doppler broadening* and *collision broadening*.

- **Doppler broadening**: Because the molecules in the sample are moving randomly in each direction (i.e. undergo Brownian motion), the light emitted when transitioning from the excited state back into the ground state is either red or blue shifted. This effect gives rise to a Gaussian profile:

$$G(\nu|\nu_0,\gamma) = \frac{1}{\gamma\sqrt{2\pi}} \exp\left(\frac{-(\nu-\nu_0)^2}{2\gamma^2}\right) \tag{2.18}$$

  Where ν is the wavelength, ν_0 is the central peak wavelength, and γ is a scale factor controlling the width of the peak. The full width at half maximum (FWHM) is defined as $2\gamma\sqrt{2\ln 2}$.

- **Collision broadening**: The collisions between the molecules in the sample cause the effective lifetime in the excited state to be lowered. As a result of the uncertainty principle this causes an increase in the uncertainty of the energy of the emitted light. The effect can be described by a Lorentzian profile:

$$L(\nu|\nu_0,\gamma) = \frac{\gamma\pi^{-1}}{(\nu-\nu_0)^2+\gamma^2} \tag{2.19}$$

  The FWHM is defined as 2γ. Compared to the Gaussian profile, the Lorentzian profile falls off less sharply and has wider wings.

The measured spectral line shape is usually a combination of both broadening effects, in which case it can be represented as a Voigt function. This function is expensive to compute, and a popular approximation is an additive Gaussian-Lorentzian form known as a pseudo-Voigt function (Wertheim, Butler, West, & Buchanan, 1974), which is much easier to compute:

$$V(\nu|\nu_0,\gamma,\eta) = \eta \cdot L(\nu|\nu_0,\gamma) + (1-\eta)\cdot G(\nu|\nu_0,\gamma) \tag{2.20}$$

Here, η is a weight factor $0 \leqslant \eta \leqslant 1$. When $\eta = 0$, the shape is equal to a Gaussian profile, and when $\eta = 1$, the shape is equal to a Lorentzian profile. An example of the different spectral line shapes is shown in Figure 2.8.

2.4. MODEL EVALUATION & CONVERGENCE

After model creation and inference, the result of Bayesian analysis is the posterior distribution, which contains all information about the combination of the probabilistic model and the observed data. The posterior distribution can be used to generate predictions. To gain more insight into the probability distribution of the model parameters (i.e. the updated priors after seeing the data), or to have a measure on the similarity of the observed data and the generated predictions, or to diagnose the sampling process, various evaluation metrics and methods exist. The most common of these are discussed next.

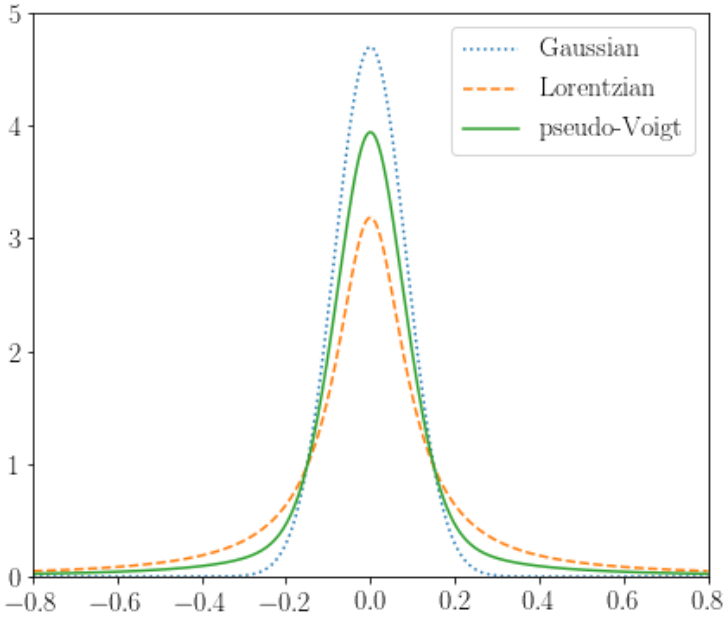


Figure 2.8: Example of the Gaussian, Lorentzian and pseudo-Voigt spectral line shapes.

2.4.1. POSTERIOR CHECKS

HIGHEST POSTERIOR DENSITY

To visualise the spread of the posterior distribution, the highest posterior density (HPD) interval is often used. The HPD is the shortest interval in which a given portion of the probability density of a model parameter is located. For example, if the 95% HPD of a parameter is calculated to lie between 3 and 4, then given the model and data, there is a probability of 0.95 that the true value of this parameter lies in this interval. By default, the HPD interval is taken as 95%, but any other value can also be used.

POSTERIOR PREDICTIVE CHECKS

As mentioned, the posterior distribution, $P(\theta|y)$, can be used to generate predictions, $\hat{y}$, based on the observed data, y, and the model parameters, θ. The **posterior predictive distribution**, $P(\hat{y}|y)$, is given by:

$$P(\hat{y}|y) = \int_{\theta} P(\hat{y}|\theta) \cdot P(\theta|y) \cdot d\theta \tag{2.21}$$

Equation (2.21) shows that the posterior predictive distribution is the distribution of predicted samples, averaged over the posterior distribution. The posterior predictive distribution is built up by generating predictions, using values for the model parameters by taking samples from the posterior distribution.

By comparing the predicted and the observed data, possible limitations in the model can be detected. Knowing the limitations of a particular model can help to improve the model (e.g. revising, simplifying or expanding), and also helps to evaluate whether the model is useful enough for its specific purpose. The process of visually or numerically comparing predicted and observed data is known as posterior predictive checks (PPC).

2.4.2. SIMPLICITY VS. ACCURACY

When constructing probabilistic models to account for observed data, usually a choice has to be made to determine which model does best in explaining the data.

One way to measure how accurately a model fits the observed data is by calculating the Bayesian version of the R^2 (R-squared) value (Gelman, Goodrich, Gabry, & Vehtari, 2019), where R^2 is defined as the ratio between explained variance, and explained variance plus residual variance. The R^2 value is defined between 0 and 1, where 1 indicates a perfect fit.

However, a higher accuracy does not automatically imply a better model, as a model with a higher accuracy usually also increases the model complexity (i.e. more parameters are needed to describe the model).

The potential danger of a too complex model lies in the fact that having many parameters often leads to a model which exhibits a good fit to the observed data (high accuracy), but performs poorly when accommodating new data (i.e. the model is *overfitting*). The potential danger of a too simple model is that the model becomes very inflexible, in which case it has a poor fit to the observed data (low accuracy), and the prediction performance becomes independent of new data (i.e. the model is *underfitting*).

From the discussion above it becomes clear that the most optimal situation is a model which will neither be overfitting nor underfitting the data. The trade-off between simplicity vs. accuracy should lead to a model, which will be as simple as possible, while still maintaining a level of accuracy high enough to be useful for its purpose.

2.4.3. PREDICTIVE ACCURACY

The accuracy of a model can be calculated in two ways. By measuring the *within-sample* accuracy, which uses data that has been used to fit the model, and by measuring the *out-of-sample* accuracy, which uses data that has not been used to fit the model (**predictive accuracy**). Measuring the within-sample accuracy tends to overestimate the accuracy as compared to the out-of-sample accuracy, which is why the latter is a more realistic value for the prediction performance.

As leaving out data to fit the model is often not an option, several methods have been designed to estimate out-of-sample accuracy using only within-sample data. The most prominent of these methods are *cross-validation* and *information criteria*.

- **Cross-validation**: In this strategy the data is divided into K portions. $K-1$ portions are used to fit the model and one portion is used for validation. By leaving a different portion out of the fitting set, the process of fitting and validation is then repeated for K rounds. A method which is frequently used in Bayesian analysis is Leave-one-out cross-validation (LOO) (Vehtari, Gelman, & Gabry, 2017). This is a method which approximates the results of cross-validation, without actually computing the K iterations. The lower the value of this measure, the higher the predictive accuracy of the model.

- **Information criteria**: This is a collection of strategies which all attempt to estimate the results of cross-validation. The common factor in these strategies is the use of two contributions, one which measures model fit, and one which penalizes model complexity. In Bayesian modelling, a measure known as the widely applicable information criterion (WAIC) (Watanabe, 2010) is often used.

In the definition of WAIC the two contributions are: the log pointwise predictive density, $lppd$, which accounts for the model fit, and a bias correction term, p_{WAIC}, which penalizes the model complexity (Gelman, Hwang, & Vehtari, 2014). To be comparable to other information criteria measures, WAIC is often written on the deviance scale, in which case:

$$WAIC = -2 \cdot lppd + 2 \cdot p_{WAIC} \tag{2.22}$$

In practice, both the terms in (2.22) can be estimated using posterior simulations, labelled θ^S with $s = 1, ..., S$, and by assuming that the number of samples S is large enough to capture the full posterior distribution. The value of the $lppd$ is calculated by taking the mean likelihood over the S posterior samples, taking the logarithm and summing up over all data points y_i:

$$lppd = \sum_{i=1}^{n} log\left(\frac{1}{S}\sum_{s=1}^{S} p\left(y_i|\theta^S\right)\right) \tag{2.23}$$

The bias correction term p_{WAIC} represents the number of effective parameters. The value of p_{WAIC} is calculated from the variance of the log-likelihood over the posterior samples S, and by summing up over all data points y_i:

$$p_{WAIC} = \sum_{i=1}^{n} V_{s=1}^{S}\left(logp\left(y_i|\theta^S\right)\right) \tag{2.24}$$

Where $V_{s=1}^{S}$ is the sample variance, defined as: $V_{s=1}^{S} a_s = \frac{1}{S-1}\sum_{s=1}^{S}(a_s - \hat{a})^2$. The intuition behind (2.24) is that models with a larger number of parameters (i.e. more complex models) will generally also have a larger spread in the posterior distribution, and thus p_{WAIC} will counteract against overestimated values of WAIC as calculated from the $lppd$ alone. When combining (2.23) and (2.24) with (2.22), the result reads:

$$WAIC = -2 \cdot \sum_{i=1}^{n} log\left(\frac{1}{S}\sum_{s=1}^{S} p\left(y_i|\theta^S\right)\right) + 2 \cdot \sum_{i=1}^{n} V_{s=1}^{S}\left(logp\left(y_i|\theta^S\right)\right) \tag{2.25}$$

As can be seen from equation (2.25), the higher the value of the log pointwise predictive density $lppd$ (i.e. the better the predictions agree with the data), and the smaller the variance of the log-likelihood over the samples p_{WAIC} (i.e. the less complex the model), the lower the value of WAIC will become, thereby increasing the predictive accuracy of the model.

2.4.4. DIAGNOSING CONVERGENCE

When running inference using MCMC sampling algorithms such as NUTS, it can take some time before the algorithm starts sampling from the correct distribution, i.e. before convergence is reached. Although in theory convergence will be reached with an infinite amount of samples, in practice only a finite amount of samples can be drawn. Therefore, before analysing the results, one of the tasks is to diagnose whether convergence has been reached. To help aid in diagnosing convergence, several statistical tests have been designed to give hints, and to flag warnings about potential problems with the sample.

CONVERGENCE TESTS

- One indicator of convergence (or non-convergence) is the $\hat{R}$ (R-hat) statistic (Vehtari, Gelman, Simpson, Carpenter, & Bürkner, 2019). This statistic compares the variance between multiple sampling chains to the variance within each chain. When convergence has been reached, the between-chain and the within-chain variances become equal, and the value of $\hat{R}$ approaches 1. As a practical rule, if the value of $\hat{R}$ is < 1.1 then this is seen as an indication that convergence has been reached.

- A second indicator which is used in diagnosing convergence is the effective sample size (ESS) (Gelman et al., 2013). The nature of the MCMC algorithm causes the samples in the chain to be autocorrelated. The ESS statistic is an indicator of how autocorrelated the samples are. In a fully converged situation, the ESS should be about equal to the total amount of samples in the chain. For practical purpose, the value of the ESS should be > 200, and a value of 1000 - 2000 is more than sufficient.

- The third indicator which can be used for checking convergence is the Monte Carlo Standard Error (MCSE) (Flegal, Haran, & Jones, 2008). The MCSE is an estimation of the error introduced by the MCMC sampling method. As the amount of independent samples grows, the MCSE value will approach 0.

- The last statistic which is mentioned here as an aid in the diagnosis of convergence, is the Bayesian fraction of missing information (BFMI) (Betancourt, 2016). The BFMI can be understood as a measure of how efficiently the sampling process is proceeding. For fast and efficient exploration of the posterior the BFMI ≈ 1, for slow exploration the BFMI $\rightarrow 0$. For practical use, a BFMI value of < 0.3 is considered as an indication of poor sampling efficiency.

3

Related Work

A literature search was conducted to find relevant publications concerning Bayesian modelling for spectroscopic data analysis. The search was focused on articles and papers concerning *vibrational spectroscopy* only, which includes both IR and Raman spectra, as these spectroscopic techniques produce sample spectra with peak shapes which appear similar, and have a common underlying physical model which can explain the observed peak shapes (Bradley, 2007).

The results of this literature search provide insight into the question as to what kind of models are used in Bayesian modelling of spectroscopic data. It also provides information about the model priors used, and other information specific to the modeling of vibrational spectra. In Chapter 4 this knowledge is used in the design and implementation of the dataset generator and of the probabilistic model.

For peak detection and separation in Surface Enhanced Raman Spectroscopy (SERS) in the study of DNA and 6-mercapto-1-hexanol interactions, (Frøhling et al., 2016) used a Bayesian network to model the Raman spectrum on a given and limited spectral range. The model consisted of peaks, a linear baseline and Gaussian noise. The peaks were modelled as pseudo-Voigt functions (see Section 2.3). The baseline was modelled as a linear function, which was a feasible assumption because of the limited spectral window used to fit the peak. Flexible priors were chosen for the model parameters, using the gamma, normal and beta distribution. The peak-fitting analysis on 300 raw Raman spectra per sample was performed using MATLAB[1] software. The model parameters were inferred using the Metropolis-Hastings algorithm, run with 10^5 iterations and a burn-in period of $5 \cdot 10^4$ iterations.

To introduce a new method to perform multivariate calibration (MVC), a complete SERS spectrum was modelled by using a hierarchical Bayesian network (Moores et al., 2016). The model consisted of multiple peaks, which were modelled as pseudo-Voigt functions, a smoothly varying baseline, which was estimated as a penalised cubic spline, and additive white noise. Informative priors, which were estimated from the results of computational chemistry and peak fitting on experimental data, were used for peak locations and for peak shape parameters. The study used two experimental datasets. The first dataset consisted

[1] https://www.mathworks.com/

of 15 SERS spectra of four different Raman-active dye molecules, eosin, fluorescein (FAM), rhodamine B, and tetramethylrhodamine (TAMRA). The second dataset was a dilution series. It consisted of 21 different TAMRA concentrations, with 15 spectra per concentration, giving a total sample size of 315 spectra. Each spectrum was measured using a resolution of 2401 wavenumbers per spectrum. The model was implemented as an open-source software package in the R statistical computing environment[2]. To be robust against local maxima, parameter inference was performed using a sequential Monte Carlo algorithm.

More recently, (Han & Ram, 2019) introduced a two-step algorithm which uses Bayesian modelling to estimate the concentration of a single component form a complex chemical mixture. The paper builds upon earlier Bayesian modelling work by composing the Raman spectrum from peaks $f_P(\nu)$, which are modelled as pseudo-Voigt functions, a baseline signal $f_B(\nu)$, modelled as a B-spline, and a noise component ϵ, modelled as independent and identically distributed (i.i.d.) Gaussian random noise, i.e.:

$$y = f_P(\nu) + f_B(\nu) + \epsilon \tag{3.1}$$

In the first step, the hierarchical Bayesian model is used to learn the model space and model parameters using Gibbs sampling and a reversible-jump Markov chain Monte Carlo (RJMCMC) algorithm, which is a modification of the Metropolis-Hastings algorithm to allow sampling from the posterior distribution when the dimension of the model is not known (i.e. the number of Raman peaks in the spectrum is not known ahead of time). Imposing very little prior knowledge before inference, mostly uninformative priors were chosen for the model parameters. The first step of the algorithm is performed on a single reference Raman spectrum of the isolated chemical component.

In the second step, the concentration of the chemical component is estimated from a complex mixture. During this step the spectrum is modelled as composed of the peaks of the single component, $f_T(\nu)$, and the peaks originating from the other components in the mixture, $f_I(\nu)$, the baseline, $f_B(\nu)$, and noise ϵ, i.e.:

$$y = f_T(\nu) + f_I(\nu) + f_B(\nu) + \epsilon \tag{3.2}$$

Where the target signal $f_T(\nu)$ is related to the concentration of the component in the mixture c_{mix}, and to the results calculated for the single component in step one $\widetilde{f}_P(\nu)$ as:

$$f_T(\nu) = c_{mix} \cdot \widetilde{f}_P(\nu) \tag{3.3}$$

The validity of the two-step algorithm was tested using simulated spectra of the single chemical component and of the mixture. For this task, 35 test sets with a fixed number of interfering components N_I and a fixed Gaussian noise scale σ were simulated. Each test set consisted of 1000 generated spectra, were the identity of the interfering components and their concentrations was varied randomly. The performance of the algorithm was also compared to three popular chemometric techniques for multivariate regression, Partial Least Squares Regression (PLS), Principle Component Regression (PCR) and Ridge Regression (RR). It was shown that for sample sizes less than ≈15 spectra the algorithm outperformed the other methods.

[2]https://www.r-project.org/

In addition, the algorithm was further tested by directly measuring the concentration of glucose from Raman spectra obtained from an aqueous mixture taken from a biopharmaceutical process. However, before the glucose concentration determination, the raw spectral data was pre-processed by taking the mean of 10 measurements for each spectral point, applying a smoothing Savitzky–Golay filter, limiting the spectral window, and finally subtracting the water signal from the data. The paper does not mention what tools or computing environment was used in the development of the two-step algorithm.

4

Experimental Setup

The results of the literature search presented in Chapter 3 provide an insight into the existence and structure of models used in the probabilistic modelling of vibrational spectra. This information is useful in answering the first research question (see Section 1.2). The second research question focuses on systematically creating differences between the spectroscopic data and the probabilistic model (see Section 1.2), to study the effect on the inference outcome. To be able to generate simulated vibrational spectroscopic datasets in which specific model parameters can be controlled, a dataset generator was built.

This chapter begins by describing the design of the dataset generator. Next, the probabilistic model which is used in the Bayesian analysis of the data is described. The chapter ends by providing an overview of the scenarios which are used to systematically investigate the effects of model-data misalignment on the inference outcome, and shortly describes the spectral datasets which are used in the evaluation of the probabilistic model on real-world data.

4.1. Dataset Generator

Spectrum Model

In order to generate simulated vibrational spectroscopic datasets, an individual spectrum is considered to be built up from a collection of individual components, with each component contributing to the sum of the total spectrum. In picking the spectrum components, only common spectrum artefacts relevant for UV, IR and Raman spectroscopy are taken into account, e.g. baseline, scatter and noise (Engel et al., 2013). The contribution of each component to the total spectrum is modelled in a similar way as described in previous research done by (Frøhling et al., 2016; Moores et al., 2016; Han & Ram, 2019). Following this approach, a vibrational spectrum can be modelled as follows:

$$y = f_{peaks}(x) + f_{baseline}(x) + \epsilon \tag{4.1}$$

In (4.1), x is the light wavenumber, $f_{peaks}(x)$ is a function describing the total peak contribution, $f_{baseline}(x)$ is the baseline underlying the spectrum, and ϵ is the noise component, which is considered to be zero centered pure Gaussian random noise. The peak and baseline components are further described by:

$$f_{peaks}(x) = C_s \cdot \sum_{m=1}^{M} \left[A_m \cdot f_{pVoigt}(x) \right] \tag{4.2}$$

Where C_s is the light scattering constant, A_m is the peak amplitude of peak m, and f_{pVoigt} is the peakshape function, modelled as a pseudo-Voigt function (see Section 2.3):

$$f_{pVoigt}(x) = \eta \cdot \frac{\sigma_m^2}{(x-\mu_m)^2 + \sigma_m^2} + (1-\eta) \cdot e^{-\frac{(x-\mu_m)^2}{2\cdot\sigma_m^2}} \tag{4.3}$$

In (4.3), μ_m is the peak maximum of peak m, σ_m is the peak width of peak m, and η is the peakshape weight factor. When $\eta = 0$ the peakshape is Gaussian, when $\eta = 1$ the peakshape is Lorentzian.

The baseline component $f_{baseline}(x)$ in (4.1) is modelled as a linear function, in which the spectrum can have an offset and a slope:

$$f_{baseline}(x) = a_0 + a_1 \cdot x \tag{4.4}$$

PARAMETER DISTRIBUTIONS

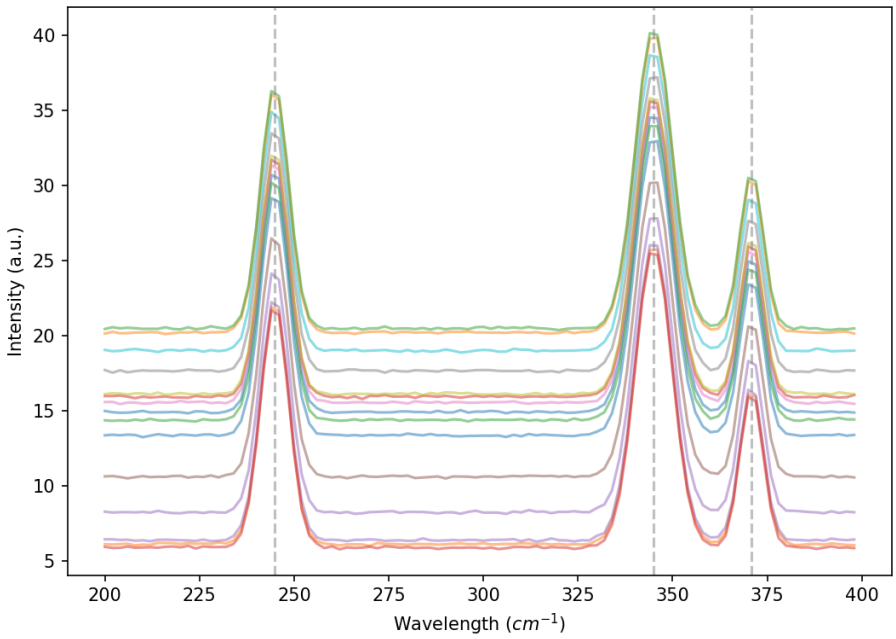
When generating the simulated datasets, the individual variable values are randomly drawn from the probability distributions shown below:

$$\begin{aligned} \mu_m &\sim U(x_{min}, x_{max}) \\ A_m &\sim U(A_{min}, A_{max}) \\ \ln(\sigma_m) &\sim \mathcal{N}(\mu_{\sigma_m}, \sigma^2_{\sigma_m}) \\ C_s &\sim U(C_{s_{min}}, C_{s_{max}}) \\ a_0 &\sim U(A_{min}, A_{max}) \\ a_1 &\sim U(0, a_{1_{max}}) \\ \eta &\sim U(0,1) \\ \epsilon &\sim \mathcal{N}(0, \sigma_\epsilon^2) \end{aligned} \tag{4.5}$$

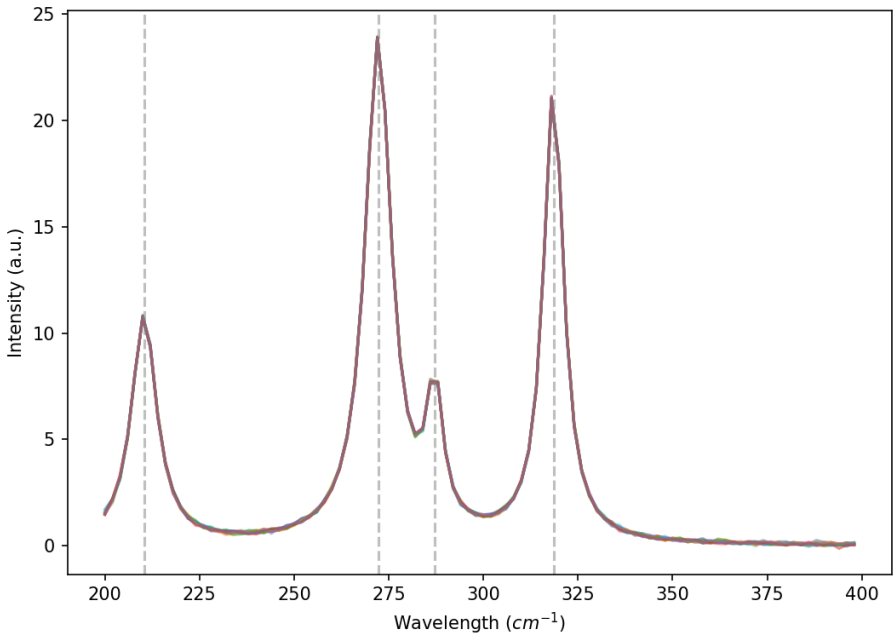
In the generation of peaks, the maximum peak location u_m is drawn from a uniform distribution between the lower and upper value of the wavenumber range, the peak amplitude A_m is drawn from a uniform distribution in a fixed amplitude range, the peak width σ_m is drawn from a log-normal distribution, and the scattering constant C_s is drawn from a uniform distribution between $C_{s_{min}}$ and $C_{s_{max}}$. The wavenumber range $(x_{min} - x_{max})$ and peak amplitude range $(A_{min} - A_{max})$ are both set by the user, and were set to $(200-400)$, and to $(5-25)$ respectively. The log-normal distribution parameter values for the peak width σ_m were taken from (Moores et al., 2016), as was measured from their spectroscopic data. The parameter values are $\mu_{\sigma_m} = 1.16$ and $\sigma_{\sigma_m} = 0.34$. For the scattering constant C_s a multiplication value of 1 ± 0.1 was taken, which is equal to the factor used by (Bjerrum, Glahder, & Skov, 2017) in creating data augmented samples for IR spectroscopy.

For a spectrum which includes an offset or slope, the spectrum offset a_0 is drawn from a uniform distribution in the amplitude range $(A_{min} - A_{max})$, and the slope a_1 is drawn from a uniform distribution between 0 and a preset maximum $(A_{max}/(x_{min} - x_{max}))$.

For all simulations, the noise component ϵ is assumed to be zero centered and normally distributed. The noise level σ_ϵ can be set by the user, and is set to 1% of A_{min} by default $(\sigma_\epsilon = 0.05)$.



(a) three Gaussian peaks ($\eta = 0$), an offset and noise (1%)



(b) four Lorentzian peaks ($\eta = 1$), no baseline and noise (1%)

Figure 4.1: Two examples of generated datasets containing fifteen spectra each. The maximum peak locations are indicated by a vertical dashed line.

EXAMPLE DATASETS

As an example, two generated datasets are shown in Figure 4.1. The first example, a dataset containing fifteen spectra, with each spectrum containing three Gaussian peaks ($\eta = 0$), an offset and noise (1%), is shown in Figure 4.1 (a). The second example, a dataset containing fifteen spectra, with each spectrum containing four Lorentzian peaks ($\eta = 1$), no baseline and noise (1%), is shown in Figure 4.1 (b). In both figures, each individual spectrum is indicated by a separate colour.

4.2. PROBABILISTIC MODEL

With the data available, the next step in a Bayesian analysis is to build a probabilistic model, and to infer the model parameters from the observed data. In the current case, which is special, the design of a model has for a large part already been done, as this model has been used to generate spectroscopic data. In fact, the probabilistic model used to analyse the generated data is exactly the same as the model used to generate the data, with the exception of some minor differences.

If the parameters in equations (4.1) to (4.4) are combined with the probability distributions shown in equation (4.5), the model can be presented as a Kruschke diagram, which is shown in Figure 4.2. The figure shows the structure of the probabilistic model that has been implemented in PyMC3, and is used to infer the model parameters from the observed data. However, there are two differences when it is compared to the model which is used in the dataset generation.

First, in the generating model, the noise level σ_ϵ is a fixed value set by the user, but in the probabilistic model, σ_ϵ is a random variable, which is to be inferred. In the probabilistic model it was chosen to use an uninformative prior for the noise level σ_ϵ, and to model this with a gamma distribution, with parameters $\alpha = 1$ and $\beta = 1$.

Second, the uniform prior on the peak locations μ_m has been relaxed with 10% of x_{min}, as compared to the range used in the generating process. The fixed range for the peak locations in the model now becomes $(0.9 \cdot x_{min} - (x_{max} + 0.1 \cdot x_{min}))$. This was done to allow for the possibility of maximum peak locations to lie outside of the observed x-range, which is used in one of the scenarios described in Section 4.3.

4.3. SCENARIO DESCRIPTION

During the initial stage of the research it became apparent that a misalignment between observed (real-world) data and the probabilistic model quickly results in a poor fit and in a poor model performance, making it unusable for further analysis. The question then arose as to how fast the process of performance degradation actually proceeds, and also how this can be quantified numerically. To help answer this question, a set of scenarios was created. The basis for these scenarios is that for every scenario there is only one single parameter difference, or misalignment, between the data generating model and the probabilistic model. In all cases, the effect of the model-data misalignment is measured by varying the parameter in the data generation process, and by keeping the probabilistic model constant.

SCENARIO SELECTION

The motivation for choosing the current set of scenarios is based on the functional form of the equation describing the spectrum (4.1), and the parts it is composed of. The form of this

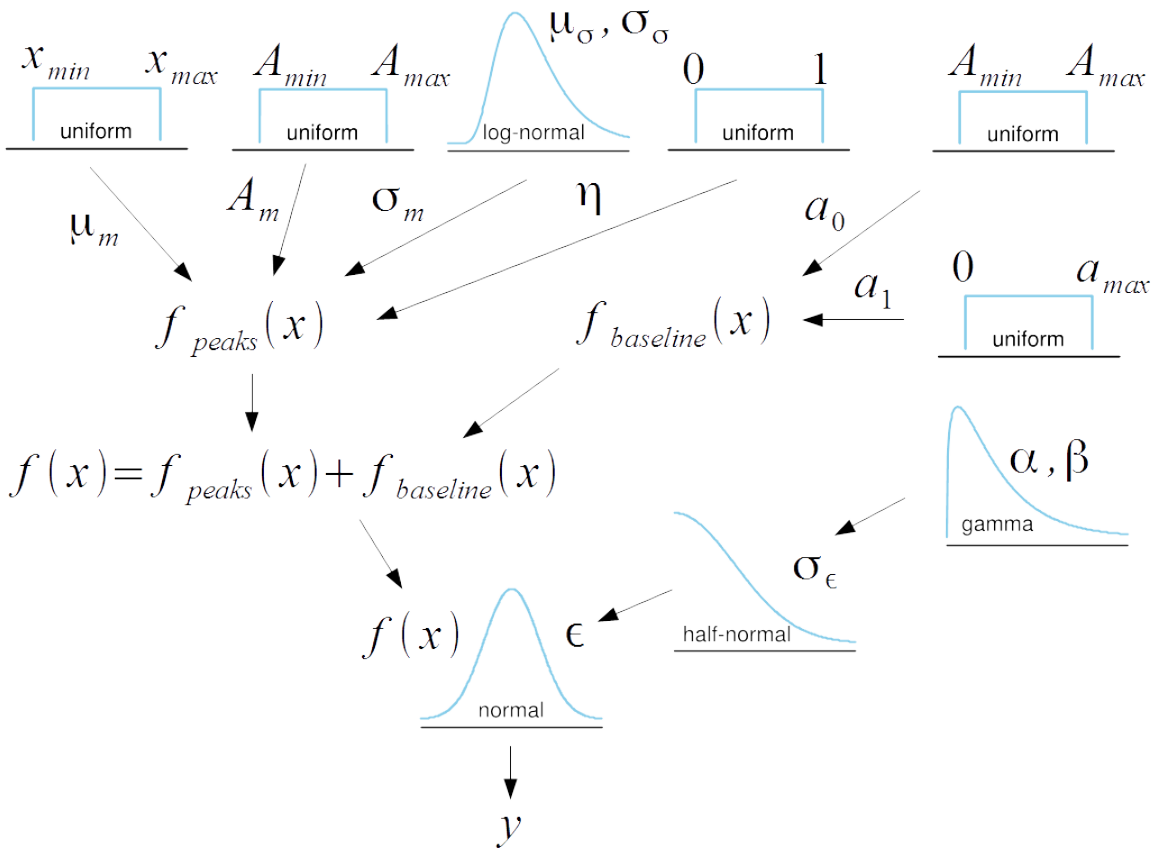


Figure 4.2: Kruschke diagram showing the parameter relations and probability distributions of the probabilistic model used for spectroscopic data analysis.

equation in turn is motivated by a study describing the artefacts which are most commonly found in real-world vibrational spectroscopic data (Engel et al., 2013), i.e. noise, baseline, and light scatter.

Equation (4.1) has three main spectrum components. The first one is the noise component ϵ. Variation of this parameter is the basis for scenario A. The second component is the spectrum baseline, which is described by equation (4.4) in this study. Variation of the parameters in this function is the basis for scenario B. The third component describes the shape and number of the spectrum peaks, which is described by equations (4.2) and (4.3). Varying the number of peaks M in the spectrum is the basis for scenario C, and varying the spectrum peak shape η is the basis for scenarios D and E. Due to the limited time availability for the present research it was chosen not to further study the influence and effect of light scatter C_s. In the last scenario, the developed probabilistic model is evaluated against three real-world datasets, this is scenario F.

4.3.1. SCENARIO A — NOISE VARIATION

In this scenario only the noise level σ_ϵ in the data is varied. The noise level is set to 1%, 2% or 5% of A_{min}, and each spectrum has three peaks ($M = 3$). Furthermore, the data in this scenario is generated with the following restrictions:

- The peaks are Gaussian ($\eta = 0$).

- The spectrum does not have a baseline ($f_{baseline}(x) = 0$).
- The light scattering constant is one ($C_s = 1$).

With these restrictions, the data generating model from (4.1) can be rewritten as:

$$y = \sum_{m=1}^{3} \left[A_m \cdot e^{-\frac{(x-\mu_m)^2}{2\cdot\sigma_m^2}} \right] + \epsilon \tag{4.6}$$

In the ideal situation, where the model used for data generation and the probabilistic model used in the inference are equal, no side-effects on the inference outcome are to be expected, i.e. the sampling convergence is always perfect, and all inferred model parameter values will be equal to the values used to generate the data. This scenario serves to test that assumption. Furthermore, it serves to test different other assumptions regarding their influence on the inference process, e.g.:

- Shape and strength of the prior;
- The initialisation settings of the inference engine used;
- Length of the sampling process;
- The number of threads used in the inference process.

4.3.2. SCENARIO B — BASELINE VARIATION

In this scenario the spectrum baseline is varied between no baseline, an offset and a linear baseline. All generated datasets have three Gaussian peaks, and a noise level of 1% ($\sigma_\epsilon = 0.05$). With the exception of the baseline, all restrictions as in scenario A apply, so rewriting equation (4.1) for this scenario gives the following model possibilities:

$$y = \sum_{m=1}^{3} \left[A_m \cdot e^{-\frac{(x-\mu_m)^2}{2\cdot\sigma_m^2}} \right] + \epsilon \tag{4.7}$$

$$y = \sum_{m=1}^{3} \left[A_m \cdot e^{-\frac{(x-\mu_m)^2}{2\cdot\sigma_m^2}} \right] + a_0 + \epsilon \tag{4.8}$$

$$y = \sum_{m=1}^{3} \left[A_m \cdot e^{-\frac{(x-\mu_m)^2}{2\cdot\sigma_m^2}} \right] + a_0 + a_1 \cdot x + \epsilon \tag{4.9}$$

With three possible baseline shapes in the data, and thus also three possibilities for the probabilistic model, there is a $3x3$ matrix of combinations between the data and the model, e.g. model (no baseline) — data (no baseline), model (linear) — data (no baseline), etc. The effect of these combinations of model and data on the sampling convergence and on the inferred model parameters is investigated in this scenario.

4.3.3. SCENARIO C — PEAK NUMBER VARIATION

In this scenario the number of peaks in the generated dataset is varied from two to six ($M = 2, ..., 6$), and the noise level is set to 1% ($\sigma_\epsilon = 0.05$). With the exception of the number of peaks, the same restrictions as in scenario A apply, and so rewriting equation (4.1) becomes:

$$y = \sum_{m=1}^{M} \left[A_m \cdot e^{-\frac{(x-\mu_m)^2}{2\cdot\sigma_m^2}} \right] + \epsilon \tag{4.10}$$

In this case there is a $5x5$ matrix of possible combinations between model and data, and the effect of these combinations on the sampling convergence and on the inferred model parameters is investigated.

4.3.4. SCENARIO D — PEAK SHAPE VARIATION I

In this scenario the spectrum peakshape η is varied in five steps from Gaussian to Lorentzian ($\eta = 0, 0.25, 0.5, 0.75, 1$). All generated datasets contain three peaks ($M = 3$) and the noise level is set to 1% ($\sigma_\epsilon = 0.05$). Rewriting equation (4.1) for this scenario gives:

$$y = \sum_{m=1}^{3} \left[A_m \cdot \left(\eta \cdot \frac{\sigma_m^2}{(x-\mu_m)^2 + \sigma_m^2} + (1-\eta) \cdot e^{-\frac{(x-\mu_m)^2}{2\cdot\sigma_m^2}} \right) \right] + \epsilon \tag{4.11}$$

There is a $5x5$ matrix of possible combinations between model and data, and again the effects of data and model misalignment on the sampling process and the inference outcome is investigated.

4.3.5. SCENARIO E — PEAK SHAPE VARIATION II

This scenario is a variation on scenario D, but instead of being set to a fixed value, here the peakshape parameter η is a free random variable in the probabilistic model. The datasets are generated with the same parameter settings as the datasets for scenario D, with three peaks per spectrum and a noise level of 1% ($\sigma_\epsilon = 0.05$). Only the peakshape η in the observed data varies ($\eta = 0, 0.25, 0.5, 0.75, 1$), and all five combinations of model and data share the same probabilistic model for the data analysis.

4.3.6. SCENARIO F — REAL-WORLD DATASETS

In this scenario the probabilistic model is applied to three real-world datasets. This scenario serves as an example to illustrate and reflect on the applicability of the developed models and modelling principles on real-world datasets. The datasets have been selected from a list of datasets that were used in the study on the use of CNNs for vibrational spectroscopic data analysis (Acquarelli et al., 2017). The following datasets are used:

- **Beers**[1]: The beers dataset contains 44 NIR spectra of Rochefort 8 (28) and Rochefort 10 (16) beers. Rochefort 8 is labelled class 1, Rochefort 10 is labelled class 2.
- **Olive oils**[2]: The olive oils dataset contains 120 FTIR spectra originating from Spain (50), Italy (34), Greece (20) and Portugal (16), corresponding to four different classes.
- **Tablets**[3]: The tablets dataset contains 310 NIR spectra obtained from four different types of pharmaceutical tablets with a varying amount of active substance. 310 samples of type A (70), B (80), C (80) and D (80).

[1] https://github.com/jnispen/PPSDA
[2] https://csr.quadram.ac.uk/example-datasets-for-download
[3] http://www.models.life.ku.dk/Tablets

5

Results and Discussion

This chapter presents and discusses the results of the experimental scenarios A-F, which are described in Chapter 4. The most relevant figures and tables supporting the results have been included in this chapter, but when applicable the reader is referred to the appendix, where supplementary material has been added.

Software Development Environment

All software created during this research has been written in the Python programming language. To support data manipulation, numerical calculation, probabilistic model definition, parameter inference, and results visualisation, various Python libraries were used. The most important libraries are: Pandas[1], Numpy[2], PyMC3[3], Arviz[4] and Matplotlib[5]. The complete workflow of data loading or data generation, data manipulation, probabilistic model definition, parameter inference, and results visualisation is bound together in a collection of interactive Jupyter notebooks[6]. The project is hosted as a GitHub project, and can be downloaded from: `https://github.com/jnispen/PPSDA`.

Runtime Environment

Depending on the expected time to complete a particular experiment, all experiments were performed either on a local PC with a virtual machine running a Linux distribution (Ubuntu 18.04.4 LTS, 16GB/128GB SSD/Intel(R) Core(TM) i5-2400 CPU@3.10GHz x 4), or on two Ubuntu Linux based virtual cloud servers (Ubuntu 18.04.4 LTS, 16GB/320GB SSD/Intel Xeon(R) Gold 6140 CPU@2.30GHz × 6).

5.1. Scenario A — Noise variation

In this scenario the noise level σ_ϵ in the data is varied. As the probabilistic model and the generating process are nearly equal, it is expected that parameter inference should be relatively straightforward (see Section 4.3.1). Therefore, the influence on inference outcome

[1] https://pandas.pydata.org/
[2] https://numpy.org/
[3] https://docs.pymc.io/
[4] https://arviz-devs.github.io/arviz/
[5] https://matplotlib.org/
[6] https://jupyter.org/

of variations in other process parameters (e.g. priors, algorithm parameters) can more systematically be investigated.

DATASETS

The noise level σ_ϵ in the generated data is varied in three steps, from 1%, 2% to 5% of A_{min} ($\sigma_\epsilon = 0.05, 0.10, 0, 25$). All generated spectra contain three Gaussian peaks ($\eta = 0$), no baseline, and each dataset contains fifteen generated spectra. Per noise level ten datasets were generated, creating a total of 30 unique datasets for this scenario.

INFERENCE SETTINGS

Parameter inference was performed using the NUTS sampler (Hoffman & Gelman, 2014), using two independent chains, taking 2000 samples per chain, and using 500 samples for tuning (burn-in). The NUTS sampler was initialised using the *adapt_diag* initialization method.

RESULTS ON STANDARD MODEL (LOGNORMAL MODEL)

To illustrate the convergence results using the standard model (Eq. 4.6), Figure 5.1 presents some example plots of generated datasets for noise levels σ_ϵ of 1%, 2% and 5% together with the posterior samples and the 95% HPD interval. As the figure shows, the posterior samples nearly perfectly capture the shape of the generated datasets, for all three noise levels σ_ϵ. The various convergence and evaluation statistics (see Section 2.4) which can be calculated from the inference results are summarized in Table 5.1.

Table 5.1: Convergence and evaluation measures for the datasets shown in Figure 5.1.

σ_ϵ **(real)**	$\hat{R}$	R^2	**WAIC**	**MCSE**	**ESS**	**BFMI**	σ_ϵ
0.05	1.0	0.9999	-4731.52	0.0000	3180.5	1.0016	0.0498
0.10	1.0	0.9996	-2633.32	0.0006	2608.3	1.0428	0.1003
0.25	1.0	0.9967	126.65	0.0005	3708.0	1.1157	0.2516

The sampling convergence $\hat{R}$ is perfect for all three noise levels, with low sampling errors levels (MCSE), a good sampling efficiency (BFMI) and a high number of effective samples (ESS). As expected, the model fit R^2 is nearly perfect, and the predictive accuracy, as expressed by the WAIC decreases with lower noise levels. In all three cases, the inferred noise level σ_ϵ is equal to the real noise level, when rounded off to the second decimal.

The results shown above represent the ideal situation, where the probabilistic model used in the inference of the model parameters is a near exact copy of the model which is used in the generation of the data (see Section 4.2). In reality there normally is little to no control over the data from which to infer the model parameters, and the exact form and shape of the data generating process is generally hidden or too complex to model. It is therefore very unlikely that there will be an exact alignment between the model and the data. Also, the various calculation methods used to infer model parameters are know to work well on some classes of problems, but not on others, making the choice for the optimal combination of model and inference algorithm for the observed data even harder. The next paragraphs will present the results of the effect of some induced misalignments between model, sampler and data which were investigated.

EFFECT OF PEAK LOCATION INFORMATION AND INIT METHOD

The effect of having prior knowledge of the peak locations on the inference outcome was investigated by shifting the maximum peak location information by 0%, 2%, 5%, 10%, or by providing no prior peak information at all. Prior peak location information is passed to the model before inference and used as a hint for model initialisation[7]. Parameter inference on the 30 generated datasets was run eight times and averaged, as to exclude single run effects. Also, to explore the effect of inference engine initialisation, the experiment was performed using three different sampling algorithm initialization methods (*jitter+adapt_diag*, *advi+adapt_diag* and *adapt_diag*)[8].

The numerical results of these experiments have been included in the appendix in Tables A.1, A.2, A.3, A.4 and A.5. The tables show the number of times convergence was reached per initialization method and per noise level. The inference outcome was considered successful if the $\hat{R}$ value, averaged over all model parameters, was ≤ 1.1, and the R^2 value was ≥ 0.99.

Table 5.2: **LogNormal model —** Convergence score per init method and peak shift (N = 30).

	init method		
Peak shift	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
0%	10.00 (±1.58)	26.88 (±1.05)	28.38 (±0.48)
2%	5.12 (±1.45)	24.50 (±1.12)	28.25 (±0.66)
5%	4.88 (±1.69)	20.88 (±1.65)	23.12 (±1.90)
10%	4.12 (±1.36)	8.88 (±1.36)	10.00 (±2.45)
no info	1.50 (±1.50)	3.75 (±1.30)	2.50 (±1.32)

The aggregated and averaged results for all noise levels and eight inference runs, have been summarized in Table 5.2. As can be seen from Table 5.2, going from exact peak location information (peak shift = 0%) to no peak location information, the convergence score decreases quite rapidly for all three initialization methods. Also, the *jitter+adapt_diag* init setting, which is currently the default setting for the NUTS sampler in PyMC3, is outperformed by the other two methods in all cases where peak location information is provided. The performance of the *advi+adapt_diag* init setting is slightly behind the *adapt_diag* init setting, which seems to performs best in all cases where location information is provided. This method was chosen as the default initialization method for the rest of the experiments in all scenarios. It is also clear from these experiments that without providing any peak location information, sampling convergence is very low (for any initialization method), even though the probabilistic model, the model parameter priors and data generating process are fully aligned.

EFFECT OF PRIOR SHAPE

Next to providing prior information with different strengths, another induced difference which was investigated was the parameter prior shape. In this experiment, the prior for the maximum peak location μ and peak width σ were modelled as normally distributed with relatively high, or flat, standard deviation ($\sigma_{mu} = 50$ and $\sigma_{sigma} = 100$). In the text this

[7] https://docs.pymc.io/notebooks/api_quickstart.html
[8] https://docs.pymc.io/api/inference.html

model is referred to as the Normal model (as opposed to the LogNormal model, which is the default model).

Two experiments were performed, one with, and one without providing peak location information before parameter inference. The results are included in the appendix in Tables A.6 and A.7. When aggregated for all noise levels, and averaged over eight inference runs, the following results are obtained, Table 5.3:

Table 5.3: **Normal model —** Convergence score per init method and peak shift (N = 30).

	init method		
Peak shift	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
0%	11.12 (±2.52)	14.12 (±1.16)	14.38 (±1.11)
no info	2.25 (±1.39)	3.62 (±1.11)	2.50 (±1.22)

From looking at the convergence score in Table 5.3, it is clear that even though the exact peak location is provided, in less than half of the 30 datasets the inference process has lead to a successful convergence, which is a large difference with the situation where model and data were in full alignment (LogNormal model). When looking at the differences between the three initialization methods, the difference in convergence score is much smaller, and no obvious 'best method' can be pointed out for this model, although *advi+adapt_diag* and *adapt_diag* perform better than *jitter+adapt_diag*.

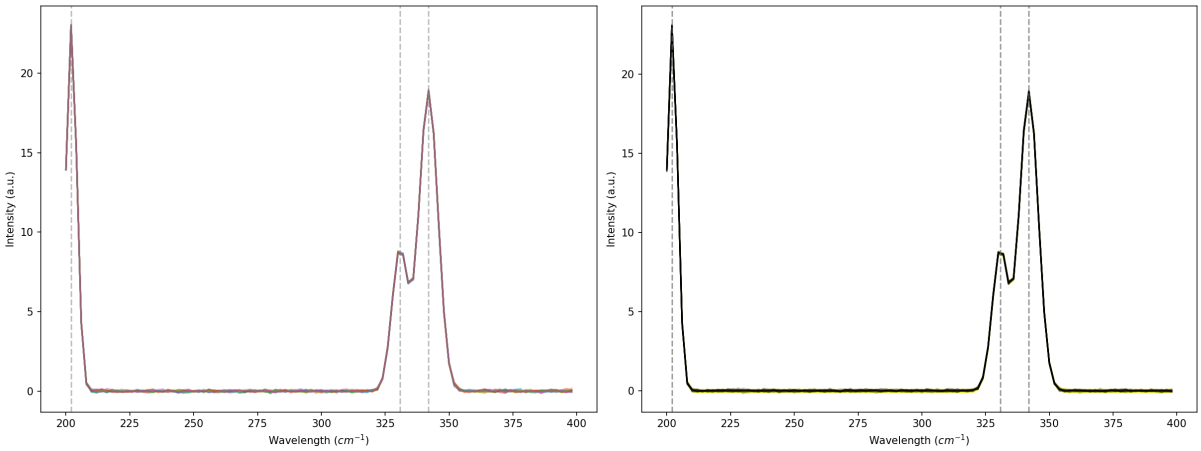
The results of experiments A.1 to A.7 can be summarized over the inference runs by noise level, and shown graphically in a barplot per init method. As no significant differences per noise level were noticed, the plots have been added to the appendix, for reference only. The plots are shown in Figures A.1 to A.7.

EFFECT OF NUMBER OF INDEPENDENT THREADS

All experiments described so far were performed by running two independent threads in parallel. The effect of running a larger number of threads in parallel on the inference outcome was investigated by running the experiments on a hexacore-CPU, and by setting the *cores* variable to six in the sampler initialization settings. The experiments were run four times for each noise level, for both the LogNormal and the Normal model, and by providing exact and no peak location information before inference. The results of these experiments have been added to the appendix as Tables A.8, A.9, A.10 and A.11. Looking at the results for the LogNormal model (Table A.8), the effect is that the convergence score for the initialization method *adapt_diag* remains on the same level, and that the score for *advi+adapt_diag* is slightly lower, while the score for the method *jitter+adapt_diag* decreases significantly. When no peak information is provided (Table A.9), the resulting convergence score is reduced to zero. The same trend in convergence score is observed for the Normal model (Tables A.10 and A.11).

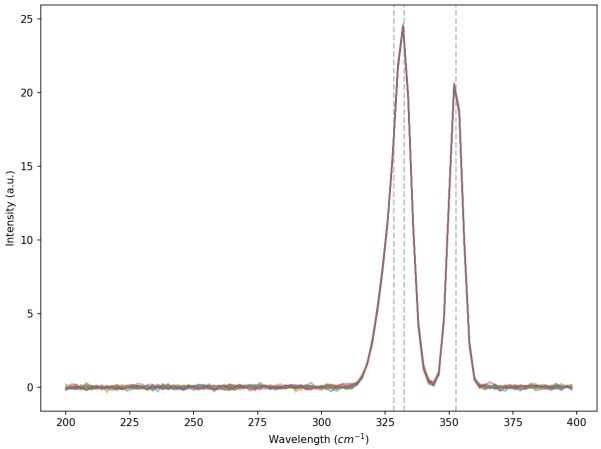
EFFECT OF NUMBER OF SAMPLES

The last experiment performed for scenario A was to increase the number of samples. For both the LogNormal and Normal model, using the *adapt_diag* init setting and providing no peak location information, parameter inference was run using 20000 samples per thread. However, this setting did not significantly change or improve the convergence result, as compared to the convergence score using 2000 samples.

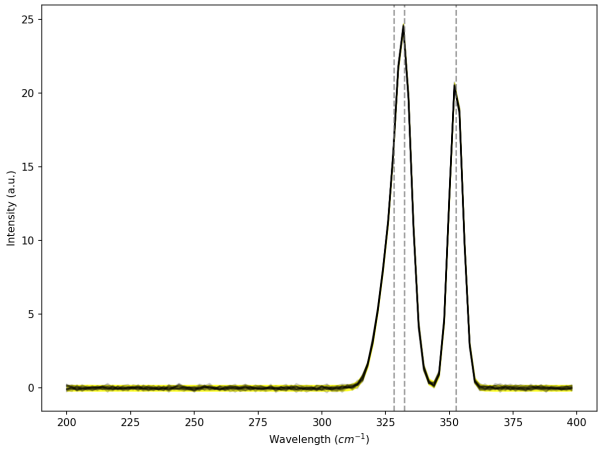


(a) 1% ($\sigma_\epsilon = 0.05$)

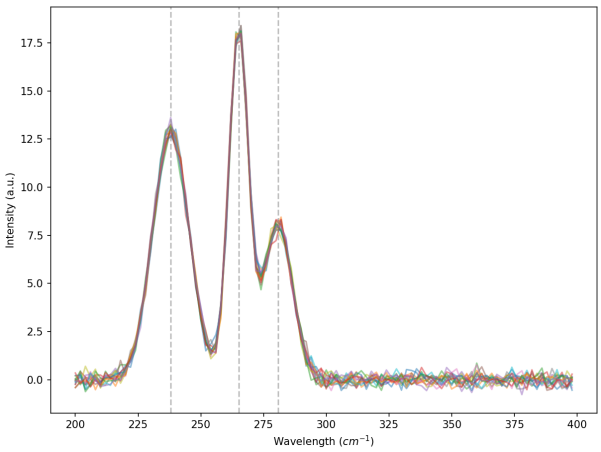
(b) posterior samples of (a)



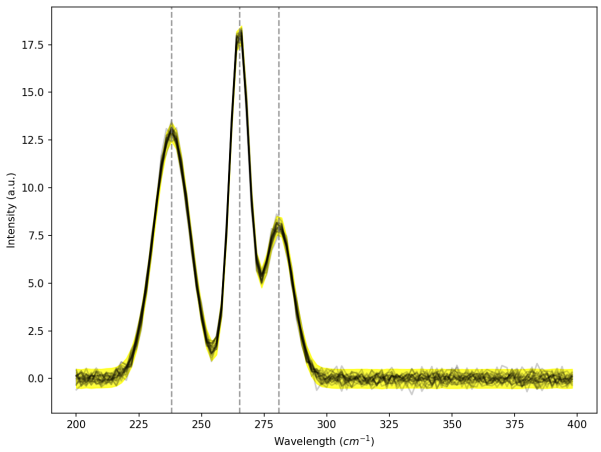
(c) 2% ($\sigma_\epsilon = 0.10$)



(d) posterior samples of (c)



(e) 5% ($\sigma_\epsilon = 0.25$)



(f) posterior samples of (e)

Figure 5.1: Examples of generated datasets for various noise levels σ_ϵ and corresponding posterior samples. The maximum peak locations are indicated by a vertical dashed line. The yellow area represents the 95% HPD interval.

5.2. SCENARIO B — BASELINE VARIATION

By varying the baseline component in the generated data, the effect on inference outcome is investigated in this scenario. It was chosen to model variations of a linear baseline, with no baseline, an offset and a linear baseline present in the data (see Section 4.3.2).

By varying the combination of model-data with regard to the baseline expected by the model and the baseline present in the data, three distinct model-data situations may occur. In the first situation the data contains a baseline that cannot be properly described by the model, i.e. the model has too few parameters and the situation is under-specified. In the second situation, the data contains a baseline that can be described with fewer parameters than the number the model has, i.e. the situation is over-specified. In the third situation the number of parameters describing the baseline in the model is equal to the number of parameters describing the baseline in the data, and the situation is called balanced.

DATASETS

Datasets containing no baseline, an offset, and a linear baseline were generated. Per baseline option four datasets were generated, with each spectrum containing three Gaussian peaks ($\eta = 0$), and a noise level of 1% ($\sigma_\epsilon = 0.05$). Each dataset contains fifteen generated spectra. Parameter inference was run eight times per model-data combination, and before each iteration new datasets were generated. This results in a total of $8 \cdot 4 = 32$ unique datasets for each model-data combination for which the evaluation measures were calculated.

INFERENCE SETTINGS

Parameter inference was performed using equal settings as for scenario A (NUTS sampler, two independent chains, 2000 samples per chain, 500 tuning samples, and algorithm initialisation using *adapt_diag*).

RESULTS

The results for this experiment are summarised in Figure 5.2. Looking at Figure 5.2 (a), the convergence score $\hat{R}$ is good ($\hat{R} \leq 1.1$) for all situations where the model is able to adapt the model parameters to the data, i.e. in situations where the model is either balanced (the diagonal in the matrix) or over-specified (the lower left triangle in the matrix). In situations where the model is under-specified (the upper right triangle in the matrix), it cannot adapt the parameters to capture the data and consequently convergence is far from ideal. This trend is reflected in the model fit R^2, which is shown in Figure 5.2 (b). A good fit is obtained in either balanced or over-specified situations, but a poor fit is seen in situations where the model cannot adapt the parameters to capture the data. In these under-specified situations also high noise levels σ_ϵ, Figure 5.2 (c), far from the real level of 0.05, and high WAIC scores are found, Figure 5.2 (d).

For reference, the heatmaps for the evaluation measures MCSE, ESS and BFMI have been included in the appendix as Figure A.8. From the figure it can be seen that for all situations the sampling efficiency (BFMI) and the effective sample size is good (ESS). The sampling error (MCSE) is low for both the balanced and over-specified case, but high for under-specified situations.

RESULTS FOR OVER- AND UNDER-SPECIFICATION

As was mentioned above, two regions exist in the heatmap where model and data are misaligned. The first region corresponds to the situation where there are more parameters than

the number that is needed to fully capture the data. This is known as the over-specified region. The second region is where the model has too few parameters to fully capture the data. This is know as the under-specified region. As is to be expected, the inference results that are found by the sampler in these regions differ.

Figure 5.3 shows an example of a posterior distribution that is found for an over- and under-specified model-data combination. In the case of over-specification (a), the solution is fully converged, and perfectly fits to the data. When looking into the posterior distribution of the model parameters, the value of the baseline slope a_1 is nearly zero ($\bar{a}_1 = 3.2x10^{-5}$), which is expected for datasets which only have a y-offset and no slope. In case of under-specification (b), the slope in the data is captured by increasing the noise level ($\bar{\sigma}_\epsilon = 1.77$).

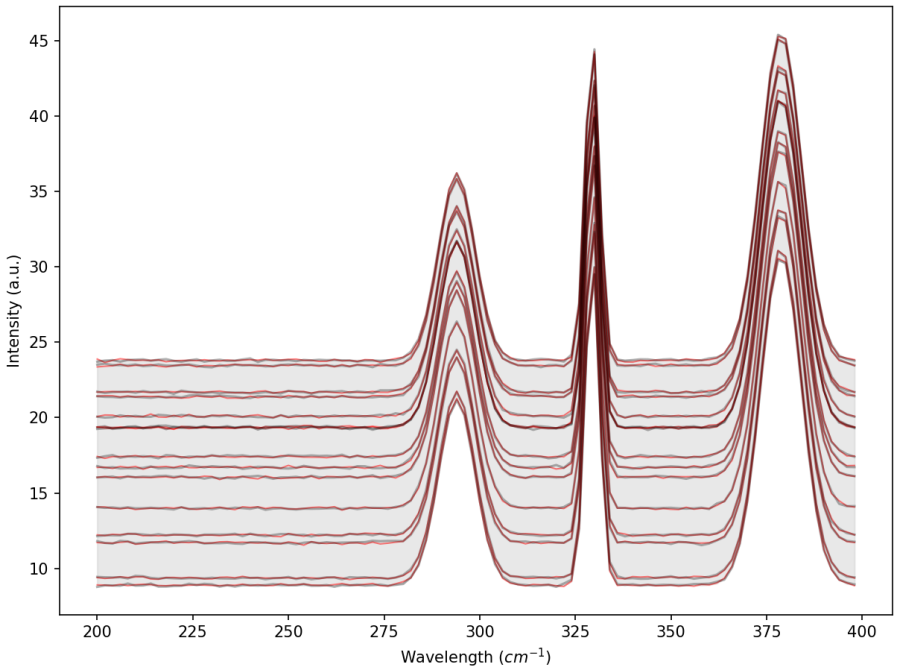
(a) R-hat ($\hat{R}$)

(b) R-squared (R^2)

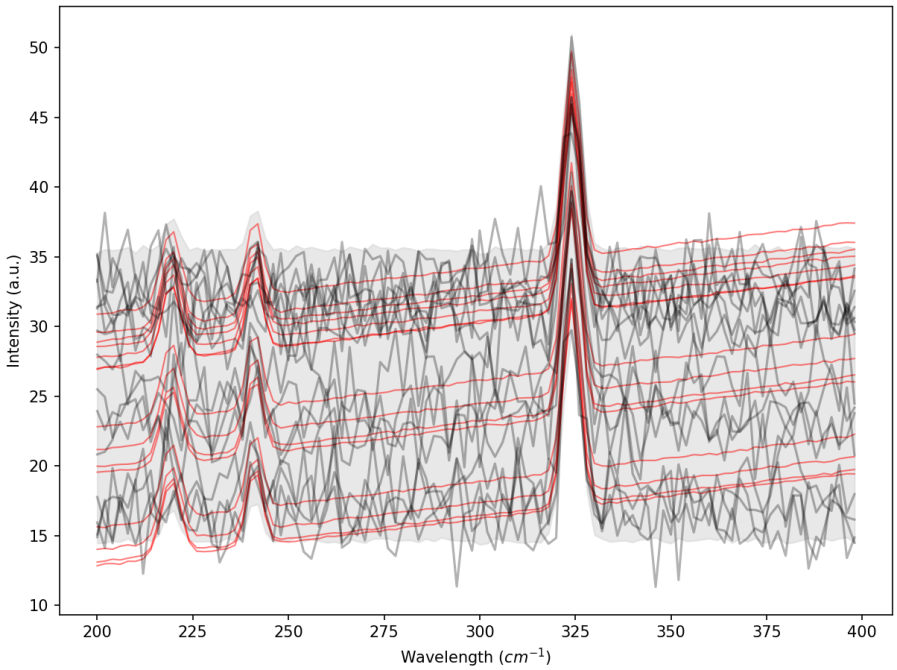
(c) Noise level (σ_ϵ)

(d) WAIC

Figure 5.2: Evaluation measures for scenario B (Baseline variation). The subfigures (a)–(d) show different evaluation measures for nine possible model-data combinations. Lower intensity color indicates a better performance of the measure. The y-axis labels refer to the probabilistic models used in parameter inference. *m_n* refers to the model with no baseline (Eq. 4.7), *m_o* refers to the model with an offset only (Eq. 4.8), and *m_l* refers to the model with a linear baseline (Eq. 4.9). The x-axis labels refer to the dataset, which is generated with a baseline according to the label name.



(a) linear model vs. offset data (over-specified)



(b) offset model vs. linear data (under-specified)

Figure 5.3: Two examples of inference outcome for an over-specified (a) and an under-specified (b) model-data combination. The generated data values are drawn as red lines, examples of posterior samples are shown as black lines. The shaded area represents the 95% HPD interval.

5.3. SCENARIO C — PEAK NUMBER VARIATION

The goal of this scenario is to study the effect on the inference outcome when the number of peaks in the generated data and the model differ (see Section 4.3.3). As was described for the model-data combinations in scenario B, also in this scenario the model-data combinations can be either under-specified, over-specified or balanced.

DATASETS

In this scenario, the number of peaks in the dataset is varied from two to six. Per peak number, four datasets were generated. Each dataset contains fifteen generated spectra, which each spectrum containing no baseline, three Gaussian peaks ($\eta = 0$), and a noise level σ_ϵ of 1% ($\sigma_\epsilon = 0.05$). Inference was run eight times per model-data combination, and before each iteration new datasets were generated. This results in a total of 32 unique datasets for each model-data combination for which the evaluation measures were calculated.

INFERENCE SETTINGS

Parameter inference was performed using equal settings as for scenario A (NUTS sampler, two independent chains, 2000 samples per chain, 500 tuning samples, and algorithm initialisation using *adapt_diag*).

RESULTS

The results for this experiment have been summarised in Figure 5.4. Reading from Figure 5.4 (a), the convergence score $\hat{R}$ is good for under-specified situations, where the number of peaks in the data is greater than what the model is expecting. In over-specified situations, where the number of peaks present in the data is lower that the number of peaks expected by the model, the convergence is not so good. On the contrast, when looking at Figure 5.4 (b), the model fit R^2 is very poor for under-specified situations, and nearly perfect for balanced and over-specified situations. The noise level σ_ϵ, Figure 5.4 (c), is high for under-specified situations, and nearly equal to the real value for balanced and over-specified situations.

The results seem to indicate that in model-data combinations where the number of adjustable parameters is greater than the number needed for a good fit to the data (over-specified), multiple solutions are possible where the model fits the data well (poor convergence, $R^2 \simeq 1$, low noise level σ_ϵ). However, in model-data combinations where the number of adjustable model parameters is too low to capture the data (under-specified), the model does not seem fit to the data very well ($R^2 \ll 1$, high noise level σ_ϵ), but a single converged solution is found ($\hat{R} \leq 1.1$). The WAIC score, Figure 5.4 (d), is high for under-specified situations, and low for over-specified situations. As for scenario B (Section 5.2), the WAIC score is following the results for R^2 and the noise level σ_ϵ, high when model fit is poor and low when model fit is good.

The heatmaps for the evaluation measures MCSE, ESS and BFMI have been included in the appendix as Figure A.9. From the figure it can be seen that in this scenario, the sampling error (MCSE), the effective sample size (ESS), and the sampling efficiency (BFMI) all follow the convergence measure $\hat{R}$, which is good for under-specified model-data combinations, and poor for over-specified model-data combinations.

The next paragraph presents an example of an inferred solution for an over- and under-specified model-data combination.

RESULTS FOR OVER- AND UNDER-SPECIFICATION

In Figure 5.5, the over-specified model-data combination of Figure 5.5 (a) is a combination where the probabilistic model expects five peaks in the data, where only two peaks are present (m_5p/d_2p). In the under-specified model-data combination Figure 5.5 (b), the data contains five peaks, where the model is expecting only two (m_2p/d_5p). The figure shows that in the case of over-specification, the probabilistic model seems to fit perfectly to the data, and in the under-specified case, the model fits to a single data peak and tries to fit all remaining peaks in the data into the second peak by increasing the noise level σ_ϵ. By looking at the posterior distributions of the model parameters this behaviour can be analysed further.

The posterior distributions of the model parameters μ, A, σ and σ_ϵ, which are shown in Figures 5.5 (a) and 5.5 (b), are included in the appendix as Figures A.10 and A.11. In both figures, ϵ corresponds to the noise level σ_ϵ.

In the over-specified case of Figure 5.5 (a), the generated data has two peaks located around wavenumbers 207 and 364. Looking at the model parameters in A.10, the inferred distribution of the first three peak locations (mu 0.0 – 0.2) is partly centred around 207-208, and partly below 200 with an arbitrary shape. Also, two of the amplitude distributions of these peaks have significant mass located around 8. The last two peaks (mu 0.3 and 0.4) have a very large portion of their location distribution centred around 364, with a major portion of the amplitude mass around 6. The net effect of these fluid location and amplitude distributions is that the shape of the observed data can be fit nearly exactly, which is reflected in the perfect scores for model fit. Because no information below wavenumber 200 (or above 400) is available from the observed data, probably any arbitrary distribution of μ (mu) in that range will be able to satisfy the constraints placed on that range. This may account for the poor convergence scores found in over-specified situations.

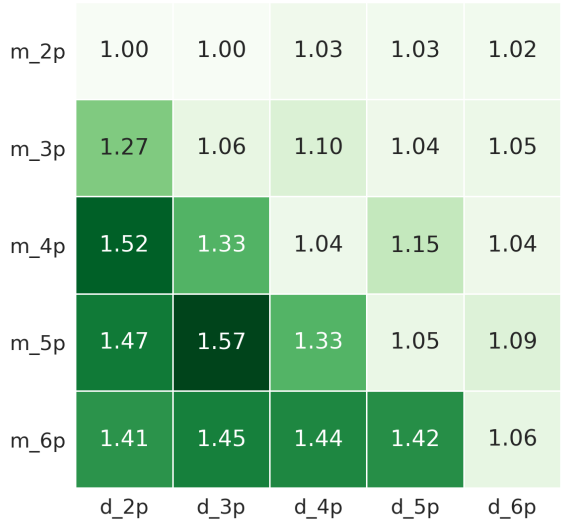
In the under-specified case of Figure 5.5 (b), the generated data has five peaks located around wavenumbers 228, 284, 295, 306 and 375. The inferred model parameter distributions presented in A.11, show two peaks (mu 0.0 and 0.1) centred around wavenumbers 228 and 293, with amplitude distributions centred around 24 and 16 respectively. Three peaks in the generated data are placed relatively close to another (284, 295, 306). In the inferred solution these three peaks all fall within the modelled location, amplitude and standard deviation of the second peak (mu 0.1) centred around 293. This is made possible by increasing the noise level (epsilon) to a sufficiently high value, so that the three closely placed peaks, for their majority, all fall within the increased noise level and second peak (mu 0.1) distribution.

The various convergence and evaluation measures for the two model-data combinations shown in Figure 5.5 and discussed above are summarized in Table 5.4.

Table 5.4: Convergence and evaluation measures for Figure 5.5 (a) and (b).

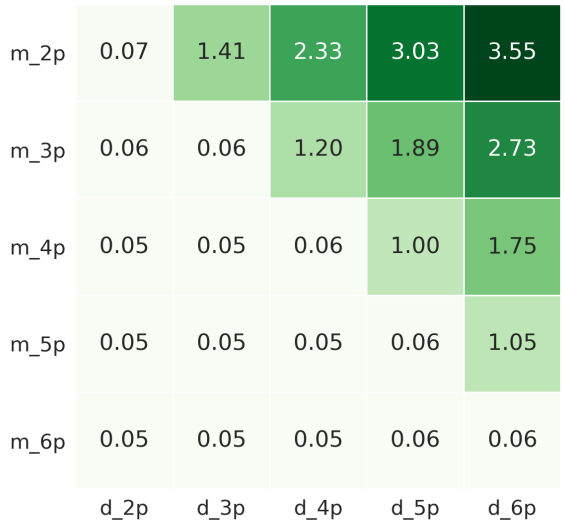
model/data	$\hat{R}$	R^2	**WAIC**	**MCSE**	**ESS**	**BFMI**	σ_ϵ
m_5p/d_2p	1.42	0.9988	-4735.07	1.7265	232.7	0.7287	0.05
m_2p/d_5p	1.00	0.8125	7088.57	0.0027	4741.6	1.0777	2.56

As expected, the evaluation measures in Table 5.4 follow the same general trend as was discussed for the values found in the model-data heatmap of this scenario.

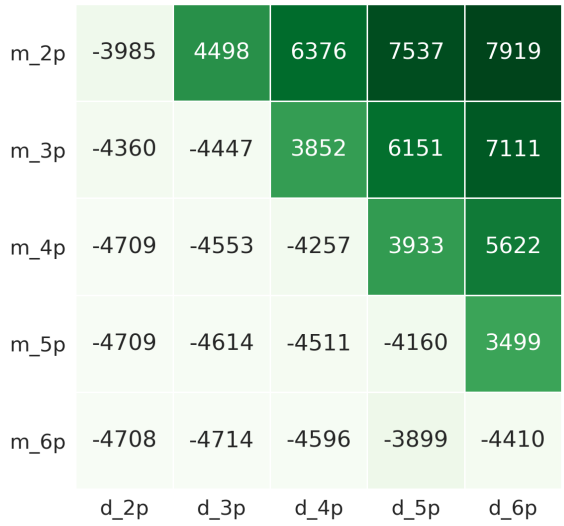


(a) R-hat ($\hat{R}$)

(b) R-squared (R^2)

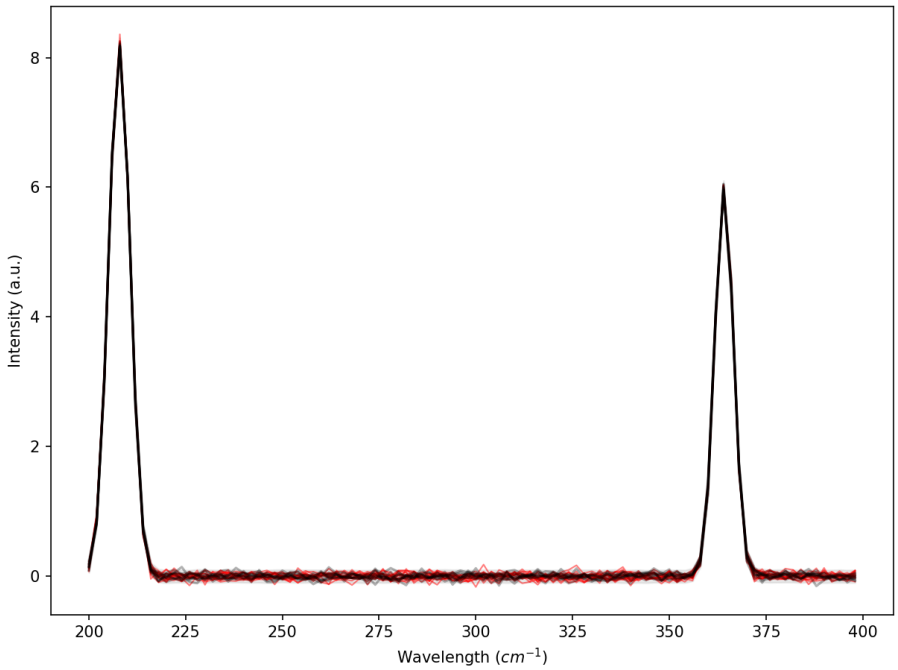


(c) Noise level (σ_ϵ)

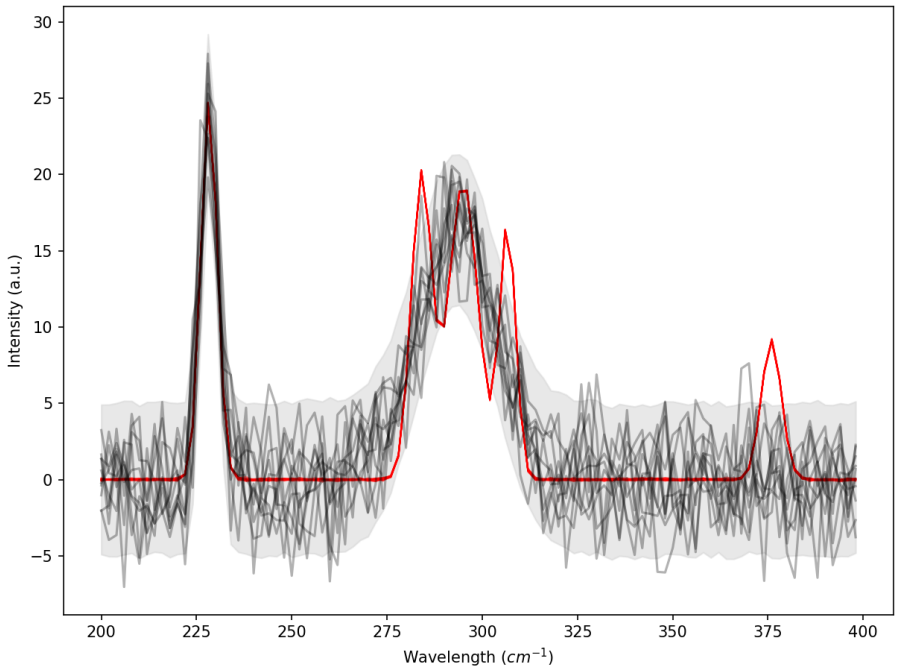


(d) WAIC

Figure 5.4: Evaluation measures for scenario C (Peak number variation). The subfigures (a)–(d) show different evaluation measures for 25 possible model-data combinations. Lower intensity color indicates a better performance of the measure. The y-axis labels refer to the number of peaks the probabilistic model is expecting to be present in the data (Eq. 4.10), and the x-axis labels refer to the number of peaks present in the dataset per spectrum.



(a) 5 peak model vs. 2 peak data (over-specified)



(b) 2 peak model vs. 5 peak data (under-specified)

Figure 5.5: Two examples of inference outcome for an over-specified (a) and an under-specified (b) model-data combination. The generated data values are drawn as red lines, examples of posterior samples are shown as black lines. The shaded area represents the 95% HPD interval.

5.4. SCENARIO D — PEAK SHAPE VARIATION I

The goal of this scenario is to systematically vary the peak shape factor η in the generated data, and to observe the effect on the inference outcome on a model with a known, predetermined, value for the peak shape factor (see Section 4.3.4).

In the probabilistic models of scenarios A-C, the peak shape factor η had been fixed to value of $\eta = 0$, resulting in datasets containing pure Gaussian peaks only. In this scenario, η will be set to other values between 0 and 1, resulting in pure Gaussian, mixed Gaussian-Lorentzian and pure Lorentzian peak shapes in the generated data.

DATASETS

The spectrum peak shape factor η is varied in five steps from Gaussian to Lorentzian ($\eta = 0, 0.25, 0.5, 0.75, 1$). Per peak shape option, four datasets were generated. Each dataset contains fifteen generated spectra, which each spectrum containing no baseline, three Gaussian peaks ($\eta = 0$), and a noise level σ_ϵ of 1% ($\sigma_\epsilon = 0.05$). Inference was run eight times per model-data combination, and before each iteration new datasets were generated. This resulted in a total of 32 unique datasets for each model-data combination for which the evaluation measures were calculated.

INFERENCE SETTINGS

Parameter inference was performed using equal settings as for scenario A (NUTS sampler, two independent chains, 2000 samples per chain, 500 tuning samples, and algorithm initialisation using *adapt_diag*).

RESULTS

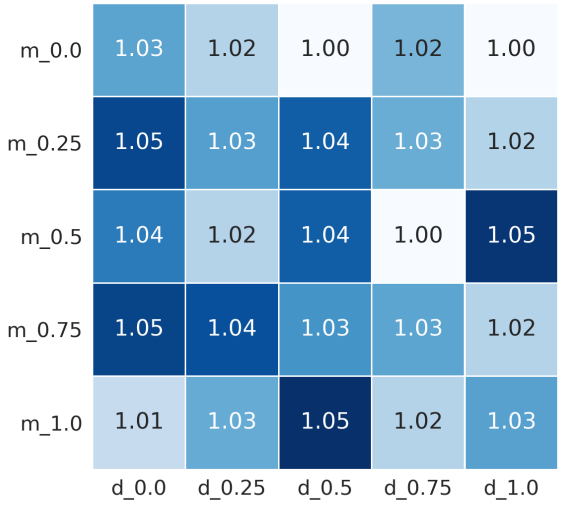
The results for this scenario have been summarised in Figure 5.6. Looking at the convergence score $\hat{R}$, Figure 5.6 (a), it is seen that for all model-data combinations convergence is good ($\hat{R} \leq 1.1$). The difference for the probabilistic model in this scenario, as compared to scenarios B and C, is the fact that all regions of the heatmap have the same number of model parameters. The value of the model parameter η differs per model, but not the number. The consequence of this is that the model only has one option to adapt to the data, and that is by varying the noise level σ_ϵ.

When looking at the model fit R^2, Figure 5.6 (b), a symmetrical result is observed, as the model and data diverge in terms of peak shape factor, the model fit decreases from a perfect fit in case of the balanced situation on the matrix diagonal, to a lesser fit in both the upper-right and bottom-left corner of the matrix, where the peak shape divergence between model and data is highest. Corresponding to a decrease in model fit in both corners, is an increase in the noise level σ_ϵ, which can be observed from Figure 5.6 (c). The WAIC score, shown in Figure 5.6 (d), as a balanced indicator of model fit and model complexity follows the same trend as observed for the noise level, low where the model fits the data best, and high in the regions where the fit is not so good, e.g. in the upper-right and bottom-left corners.

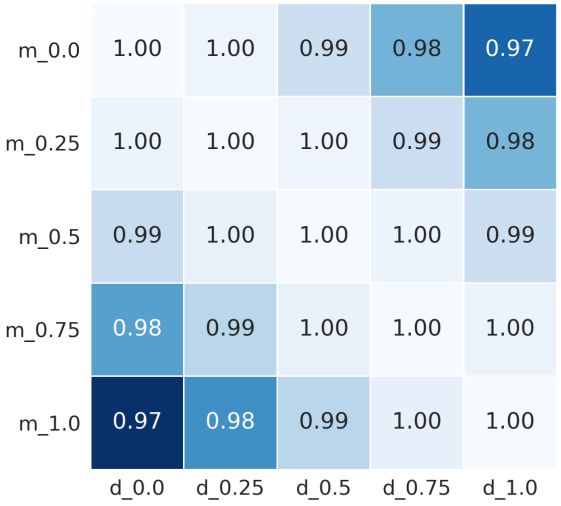
The evaluation measures MCSE, ESS and BFMI have been included in the appendix as Figure A.12. For all model-data combinations, the sampling error (MCSE) is low, the effective sample size (ESS) is high, and the sampling efficiency (BFMI) is good, all indicators of a good convergence.

An example of inference outcome for this scenario has been included as Figure 5.7. The model in this example is expecting a Gaussian peak shape ($\eta = 0$), and the data is composed

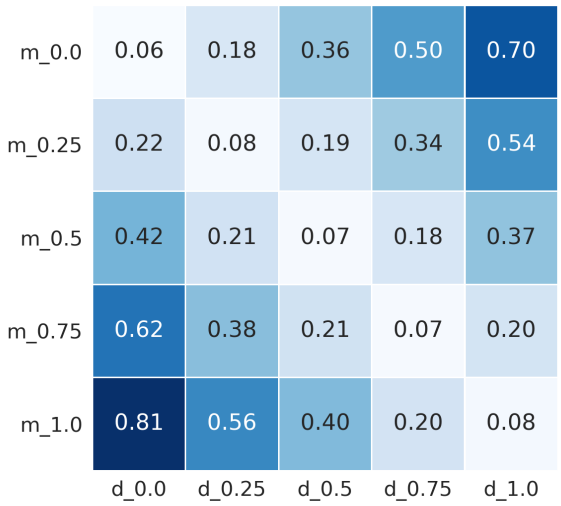
of Lorentzian peaks shapes ($\eta = 1$). As compared to Gaussian peaks, Lorentzian peaks are broader at the bottom and do not fall of as sharply as Gaussian peaks (see Section 2.3). Figure 5.7 shows that by increasing the noise level ($\sigma_\epsilon = 1.10$) the probabilistic model is able to compensate for this misalignment in peak shape factor, and converges ($\hat{R} = 1.0$).



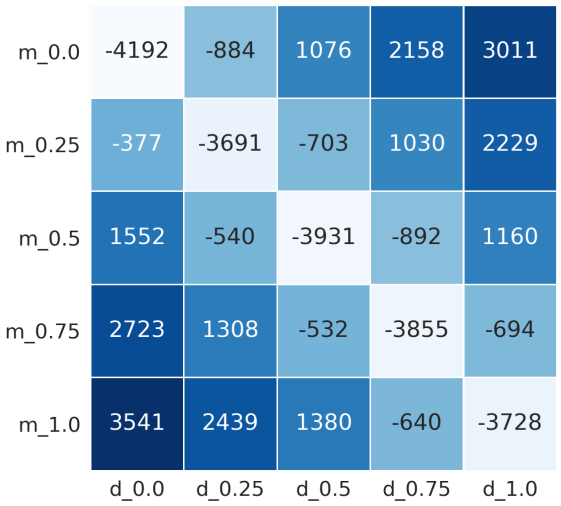
(a) R-hat ($\hat{R}$)



(b) R-squared (R^2)



(c) Noise level (σ_ϵ)



(d) WAIC

Figure 5.6: Evaluation measures for scenario D (Peak shape variation I). The subfigures (a)–(d) show different evaluation measures for 25 possible model-data combinations. Lower intensity color indicates a better performance of the measure. The y-axis labels refer to the value of the peak shape factor η that the model is expecting to find in the data (Eq. 4.11), and the x-axis labels refer to the peak shape factor that was actually used to generate the data.

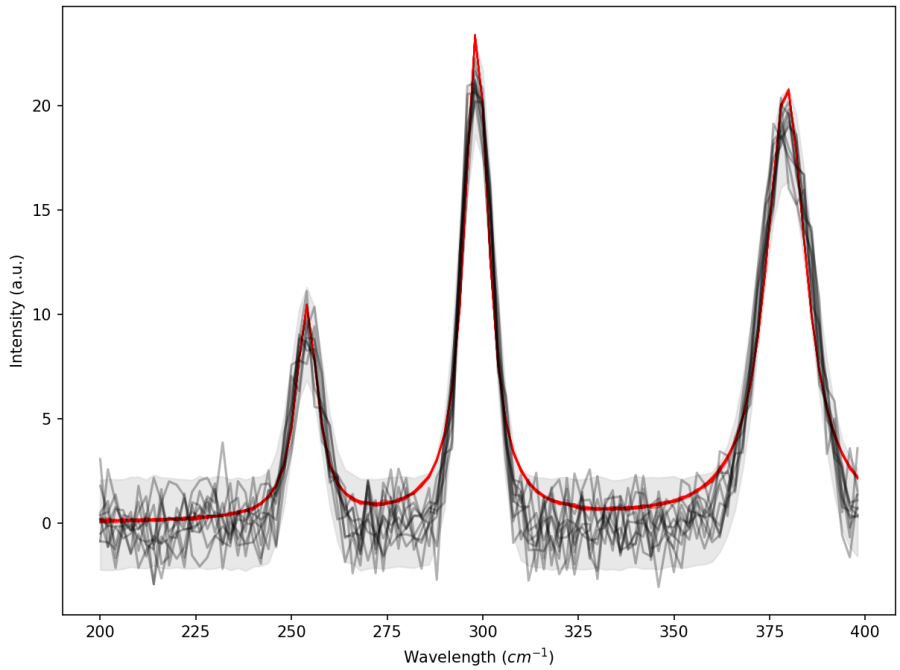


Figure 5.7: Example of a posterior plot for the combination: model (Gaussian) – data (Lorentzian). The generated data values are drawn as red lines, examples of posterior samples are shown as black lines. The shaded area represents the 95% HPD interval.

5.5. SCENARIO E — PEAK SHAPE VARIATION II

The goal of this scenario is to study the effect on the inference outcome by adding the peak shape factor η as a free random variable to the probabilistic model (see Section 4.3.5).

Although in the previous scenario, scenario D, the peak shape factor η was varied in the generated data, the value in the model was set to a fixed value. In the current model, the value of the peak shape factor is free, and is to be inferred from the observed data.

DATASETS

As was done in scenario D, in this scenario the spectrum peak shape factor η is also varied in five steps from Gaussian to Lorentzian ($\eta = 0, 0.25, 0.5, 0.75, 1$). However, instead of being set to a fixed value, the peak shape factor η is a free random variable in the probabilistic model used to infer the model parameters from the data (Eq. 4.11). Per peak shape option, 40 datasets were generated. Each dataset contains fifteen generated spectra, which each spectrum containing no baseline, three peaks, and a noise level σ_ϵ of 1% ($\sigma_\epsilon = 0.05$). In total, $5 \cdot 40 = 200$ unique datasets were generated for this experiment.

INFERENCE SETTINGS

Parameter inference was performed using equal settings as for scenario A (NUTS sampler, two independent chains, 2000 samples per chain, 500 tuning samples, and algorithm initialisation using *adapt_diag*).

RESULTS

As for scenario A, the inference outcome was considered successful if $\hat{R} \leq 1.1$, averaged over all model parameters, and $R^2 \geq 0.99$. Counted in this way, the inference outcome for 189 of the 200 datasets was successful (94,5%), which is nearly equal to the result obtained for scenario A (94,6%).

When calculated over all 40 datasets per peak shape option, the averaged convergence and evaluation measures for scenario E are summarized in Table 5.5.

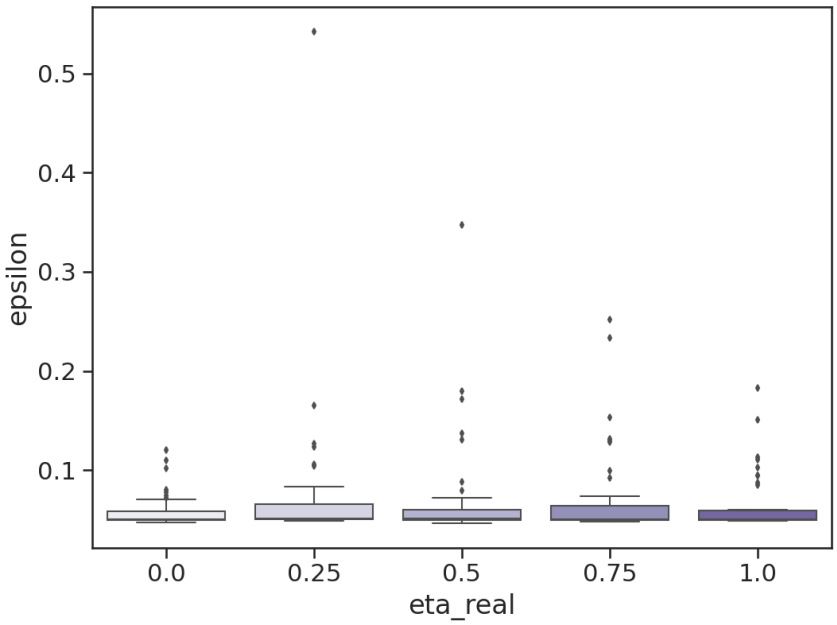
Table 5.5: Averaged convergence and evaluation measures per peak shape option η (N = 40).

η **(real)**	$\hat{R}$	R^2	**WAIC**	**MCSE**	**ESS**	**BFMI**	σ_ϵ	η
0.00	1.08	0.9997	-4342.65	0.0823	3298.4	1.0416	0.0585	0.0014
0.25	1.01	0.9995	-3972.92	0.0267	3879.6	1.0516	0.0762	0.2313
0.50	1.01	0.9995	-4069.55	0.0199	3676.4	1.0493	0.0713	0.4816
0.75	1.00	0.9988	-3637.68	0.0039	3647.7	1.0568	0.0971	0.7067
1.00	1.02	0.9996	-4145.78	0.0217	3287.5	1.0227	0.0649	0.9747

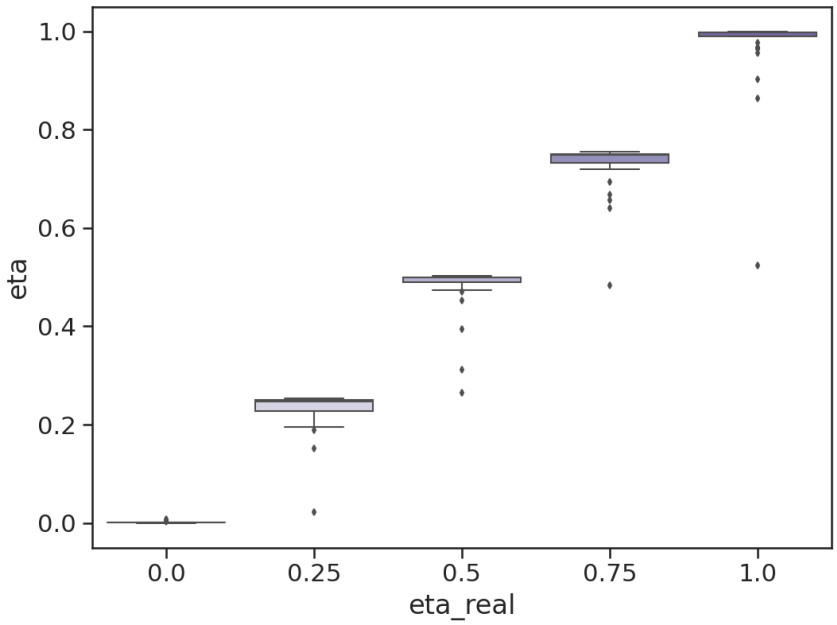
For all peak shape values, the mean convergence as measured by $\hat{R}$ is good, and the mean model fit is near perfect. Also, the other convergence indicators, MCSE, ESS, and BFMI all point to good sampling convergence. The mean noise level σ_ϵ mostly stays close to the real value, and only seems to deviate significantly for peak shape factor 0.75. The same trend is observed for η.

The values in Table 5.5 represent the mean of the 40 datasets per peak shape value, and do not consider the distribution of the inferred values for σ_ϵ and η, and also count the inference outcomes which did not converge ($\hat{R} > 1.1$). When only considering successful

inference outcomes ($\hat{R} \leq 1.1$, $R^2 \geq 0.99$), Figure 5.8 shows two boxplots which display the median, interquartile ranges and outliers for the inferred values of the noise level (epsilon) and the peak shape value (eta). Both figures show that the mass of the distribution for the inferred values of the noise level and the peak shape lie much more close to the real values than is suggested by looking at Table 5.5 alone. Figure 5.8 also shows that in spite of a perfectly converged solution, there are still quite some parameter values which lie far from the median, as depicted by the black dots.



(a) noise level (inferred) vs. peak shape (data)



(b) peak shape (inferred) vs. peak shape (data)

Figure 5.8: Boxplot of successful inference outcomes for scenario E. Median, interquartile ranges and outliers for the noise level and the peak shape value.

5.6. SCENARIO F — REAL-WORLD DATASETS

In this scenario, parameter inference is performed on real-world datasets using the developed probabilistic model. Three datasets were selected as being representative of real-world vibrational spectroscopic datasets (see Section 4.3.6).

As opposed to scenarios A-E, the exact form of the generating process, and the number and value of the model parameters are unknown. Therefore, the prior model parameter values have to be obtained indirectly, e.g. by looking at the collected data or by incorporating prior knowledge obtained from other sources.

DATASETS

The following datasets, selected from the list of datasets in the study by (Acquarelli et al., 2017), were used:

- **Beers**: The beers dataset contains NIR spectra of Rochefort 8 (class 1) and Rochefort 10 beers (class 2). Fifteen samples from class 2 were randomly drawn and used for parameter inference.
- **Olive oils**: The olive oils dataset contains FTIR spectra originating from Spain, Italy, Greece and Portugal, corresponding to four different classes. Fifteen samples from the Spain class were randomly drawn and used for parameter inference.
- **Tablets**: The tablets dataset contains NIR spectra obtained from four different types of pharmaceutical tablets with a varying amount of active substance. Fifteen samples from the type A class were randomly drawn and used for parameter inference.

INFERENCE SETTINGS

Inference was performed using equal settings as for scenario A (NUTS sampler, two independent chains, 2000 samples per chain, 500 tuning samples, and algorithm initialisation using *adapt_diag*).

RESULTS

An overview of the selected datasets, displaying fifteen random samples of each class in the dataset, is shown in Figures 5.9 and 5.10. In Figure 5.9 (a) a plot of the beers dataset is shown, Figure 5.9 (b) shows a plot of the olive oils dataset, and in Figure 5.10 (a) a plot of the tablets dataset is shown.

In all three plots shown, the difference between the classes is very hard to notice, if noticeable at all. In Figure 5.9 (a), the two classes present in the beers dataset are shown as black and blue lines. Both classes are nearly fully intermixed and can hardly be seen to differ over the whole range of the spectrum. In Figure 5.9 (b), the four classes in the olive oils dataset are show as black, blue, red and green lines. Also in this figure, the lines of the different classes nearly fully overlap, and no clear feature separating the classes can be noticed. In Figure 5.10 (a), the four classes of tablets are shown as black, blue, red and green lines. The figure shows that the classes appear to have a different distribution for the offset, with the blue class more concentrated towards lower intensities, the green class towards higher intensities, and the red and black classes more or less overlapping in intensity level. However, no feature distinguishing the classes in terms of maximum peak location or peak amplitude can clearly be seen from the figure.

To create a situation which is comparable to the inference process in scenarios A-E, this scenario draws fifteen random samples from one class in the dataset before running parameter inference. The results are shown in Figure 5.11. In each subfigure (a), (c) and (e), the fifteen uniclass samples have each been plotted in a different color.

- **Beers**: The beers dataset was selected to test the probabilistic model with a linear baseline (Eq. 4.9), as the samples in Figure 5.11 (a) appear to differ by having an offset and a slope combined. The maximum peak locations in this spectrum were estimated directly by looking at the plot, and have been added to Figure 5.11 (a) as vertical dashed lines. A plot of the posterior samples, and 95% HPD is given in Figure 5.11 (b).

- **Olive oils**: In the olive oils dataset, which is shown in Figure 5.11 (c), no offset or slope appears to be present, and the probabilistic model with no baseline was used for parameter inference (Eq. 4.7). The spectrum shows a large number of peaks. The peaks which are characteristic for the different functional bonds in olive oil were obtained from a study on the authentication of extra virgin olive oil (Lerma-García, Ramis-Ramos, Herrero-Martínez, & Simó-Alfonso, 2010), and are plotted as vertical dashed lines in Figure 5.11 (c). A plot of the posterior samples, and 95% HPD is given in Figure 5.11 (d).

- **Tablets**: The tablets dataset is shown in Figure 5.11 (e). The spectra in this plot appear to differ by an offset, and the probabilistic model with an offset baseline was used for parameter inference (Eq. 4.8). Although the study describing this dataset mentions the maximum peak location of the active substance present in the tablet (8830 cm^{-1}) and the excipient (8200 cm^{-1}), it does not mention any other peak locations (Dyrby, Engelsen, Nørgaard, Bruhn, & Lundsberg-Nielsen, 2002). Therefore, the other maximum peak locations which can be seen in the spectrum were estimated from the plot directly. The maximum peak locations are shown as vertical dashed lines in Figure 5.11 (e). A plot of the posterior samples, and 95% HPD is given in Figure 5.11 (f).

For these datasets, running parameter inference using the LogNormal model results in an exception. Re-specifying the model, e.g. by using a normal distribution as prior for the maximum peak locations μ_m, as is done in the Normal model, will remove the exception. Therefore, in this case, parameter inference was performed using the Normal model (see Section 5.1). The convergence results are summarized in Table 5.6.

Table 5.6: Convergence and evaluation measures for the real-world datasets.

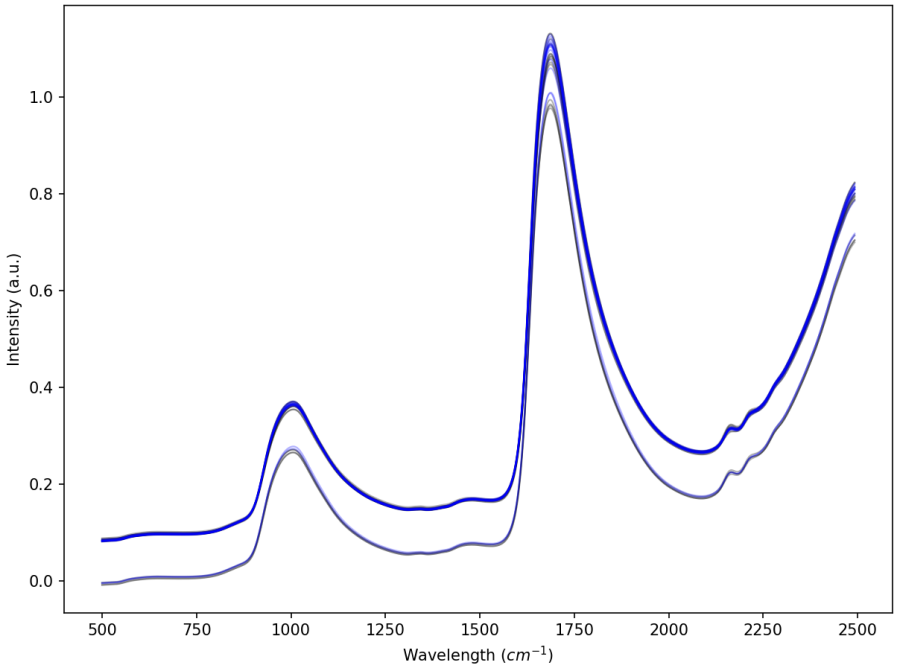
Dataset	$\hat{R}$	R^2	**WAIC**	**MCSE**	**ESS**	**BFMI**	σ_ϵ
Beers	1.38	0.9936	—	0.66054	23.0	1.1698	0.0194
Olive oils	1.90	0.9941	—	23.1529	9.4	0.2371	0.0265
Tablets	1.46	0.9871	—	15.6724	20.0	0.4859	0.0623

In none of the three cases convergence was reached, as indicated by the high value of $\hat{R}$ ($\hat{R} > 1.1$). Also, the effective sample size (ESS) is low, and the sampling error (MCSE) is high

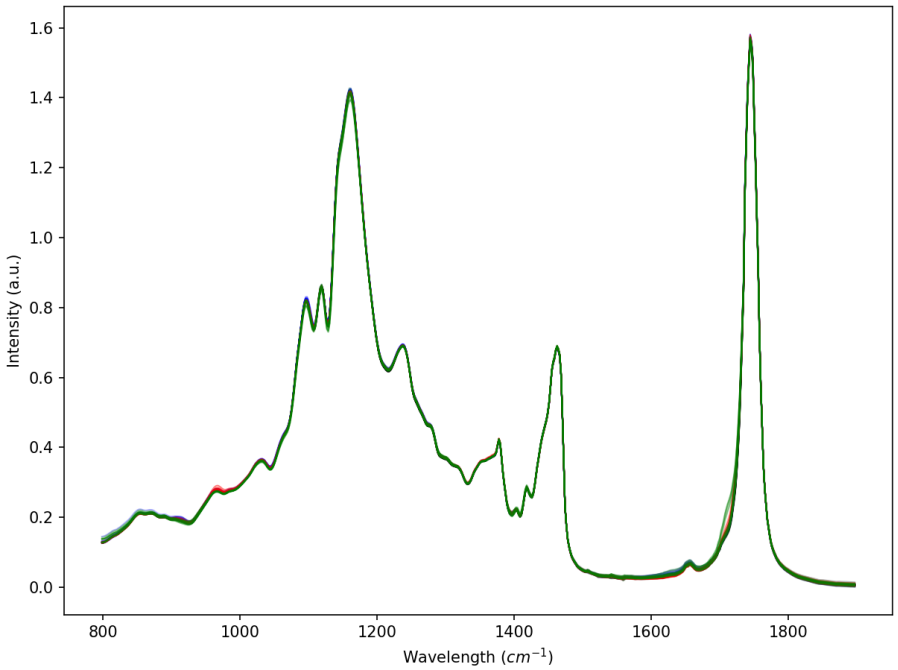
in all three cases. Particularly for olive oils, and to a minor extent for tablets, the sampling efficiency (BFMI) is low.

The WAIC values are not shown in Table 5.6, as different combinations of model and data are used for each of the three inference cases. The WAIC results are only useful when comparing different models explaining the same data, which is not the case here.

When looking at the posterior samples in Figures 5.11 (b), (d) and (f), it is seen that the general shape of the observed data is clearly present in the posterior samples, as also reflected by the measure for model fit R^2, which is always ± 0.99 when rounded off to the second decimal. However, for all three datasets the noise level σ_ϵ is much higher than the noise level present in the observed data.

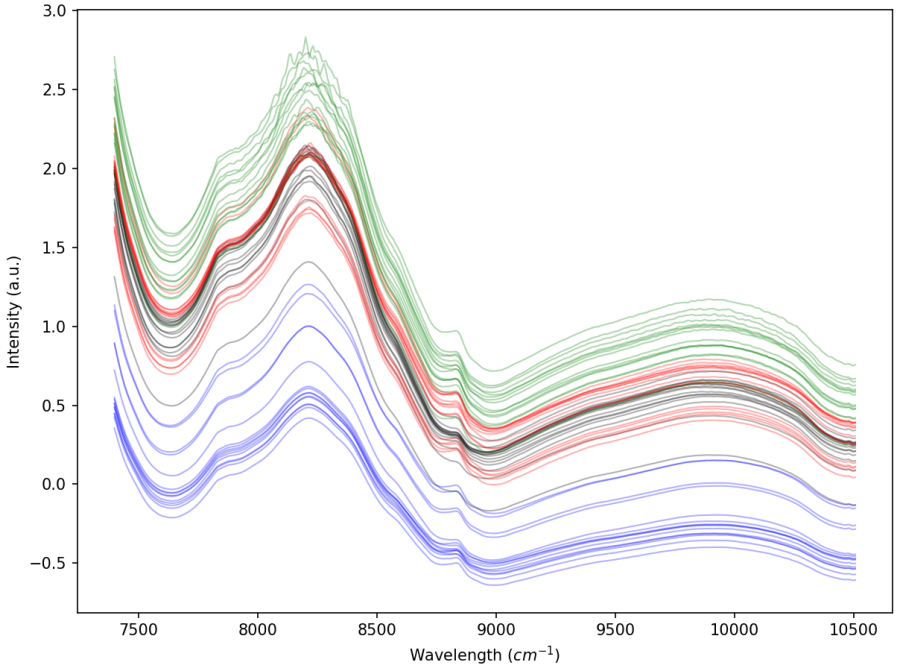


(a) Beers (2 classes)



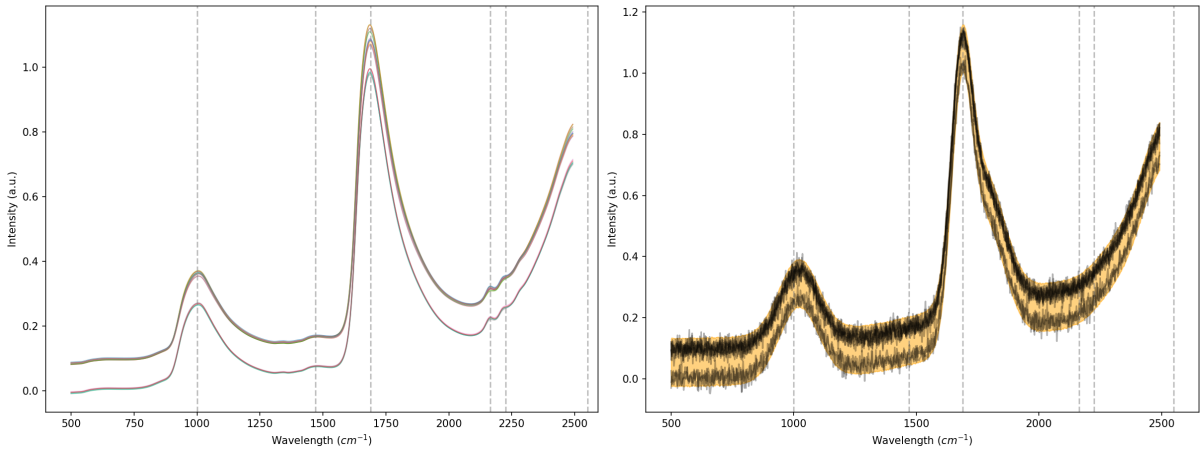
(b) Olive oils (4 classes)

Figure 5.9: Class plot for the datasets Beers and Olive oils. Each subfigure displays fifteen samples taken randomly from the classes in the datasets. Each class is drawn in a separate color: black, blue, red, or green.



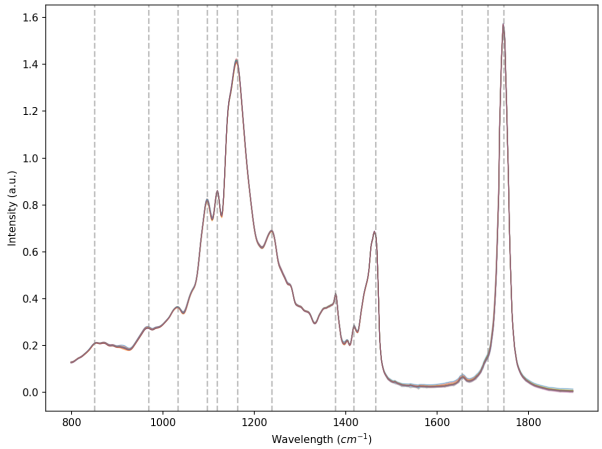
(a) Tablets (4 classes)

Figure 5.10: Class plot for the dataset Tablets. The figure displays fifteen samples taken randomly from the classes in the dataset. Each class is drawn in a separate color: black, blue, red, or green.

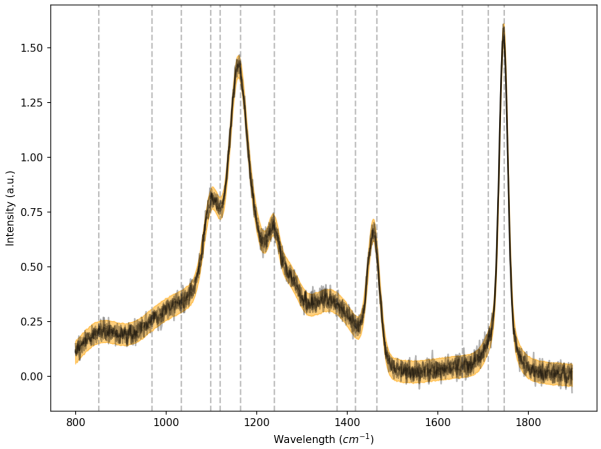


(a) Beers

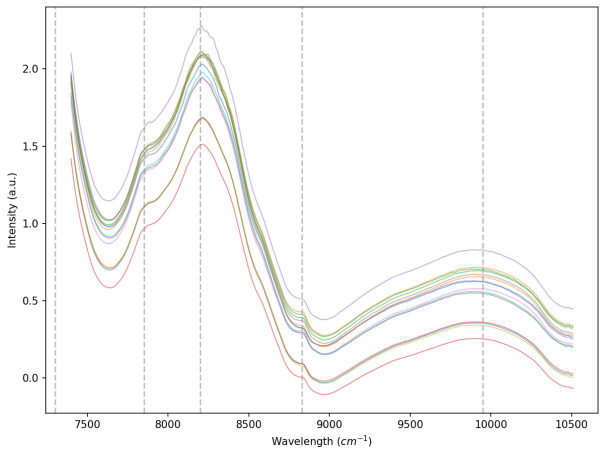
(b) Beers (posterior)



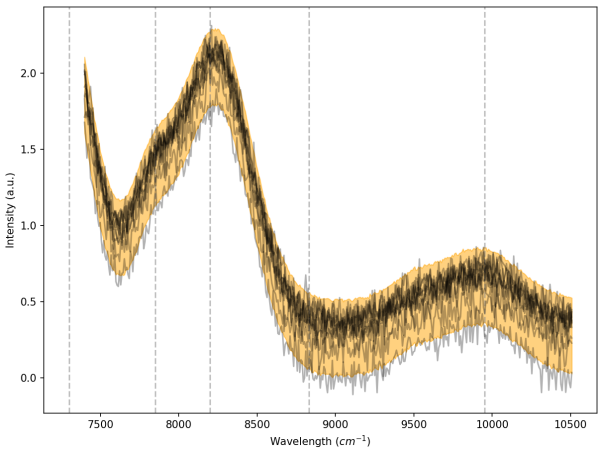
(c) Olive oils



(d) Olive oils (posterior)



(e) Tablets



(f) Tablets (posterior)

Figure 5.11: Example inference outcome for the datasets Beers, Olive oils and Tablets. The maximum peak locations are indicated by a vertical dashed line. Posterior samples are shown as black lines in the subfigures on the right. The orange area represents the 95% HPD interval.

6

CONCLUSIONS AND FUTURE WORK

In this final chapter, the conclusions which can be drawn from the research described in the preceding chapters are presented. Also, limitations with regard to the current research, and possible future work will briefly be discussed. This chapter will start with presenting the answers to the research questions, which formed the original motivation for the research, and which were described in the introduction (see Section 1.3).

6.1. CONCLUSIONS

To help focus the research effort, the introduction presented two research questions which support in answering the main research objective. The answers to the research questions and research objective are given below:

RQ1: *Can a probabilistic model be constructed that captures the characteristic features of a vibrational spectrum?*

The characteristic features of a vibrational spectrum are formed by the individual contribution of specific peaks, originating from the vibrational modes of different functional groups in the observed substance, to the total spectrum (see Section 1.1). For large molecules, or mixtures of molecules these features can quickly form a complex pattern. Moreover, in real-world settings, vibrational spectra are distorted by three major sources of artefacts, i.e. baseline, light scatter, and noise (Engel et al., 2013). Earlier work done on Bayesian modelling of vibrational spectra (Frøhling et al., 2016; Moores et al., 2016; Han & Ram, 2019), provided this research a basis for the formulation of a model which allowed for the generation of artificial spectral datasets (see Section 4.1). This model was transformed into a probabilistic model, which can be used to infer parameters from observed spectral data (see Section 4.2). The probabilistic model was both used to infer parameters values from generated datasets (see Sections 5.1 to 5.5), and from real-world datasets (see Section 5.6).

In conclusion, although the model developed during this research can be used to capture broad spectral features present in a vibrational spectrum, in the current form it is unable to capture the often subtle characteristic features, which are needed for further spectroscopic data analysis. Furthermore, some major limitations regarding the current approach exist. These limitations are discussed in Section 6.2.

Table 6.1: Summary of convergence, model fit and inferred noise level for scenarios A to F.
(+) = good, (+/-) = neutral, (-) = bad

Scenario	**Variation**	**Convergence** ($\hat{R}$)	**Model fit** (R^2)	**Noise level** (σ_ϵ)
A	noise	+	+	+
B (over)	baseline shape	+	+	+
B (under)	baseline shape	-	-	-
C (over)	peak number	-	+	+
C (under)	peak number	+	-	-
D	peak shape (fixed)	+	+/-	-
E	peak shape (free)	+	+	+
F	real-world data	-	+	-

RQ2: *What is the effect of a misalignment between the probabilistic model and the spectral data on the inference outcome?*

The results of the application of the probabilistic model on real-world vibrational datasets (scenario F, see Section 5.6) reveal that although the general shape of a collection of spectra can be captured, as indicated by the measure for the model fit R^2, convergence, as indicated by $\hat{R}$ and other evaluation measures, is generally not reached. Furthermore, the inferred noise level, which represents zero centered pure Gaussian random noise, is significantly higher than the noise level that is seen in the data of the original spectra.

In order to get a better understanding of what might be causing this, the second research question focused on creating a deliberate misalignment between the observed data and the probabilistic model. In five selected scenarios A-E, the effect on parameter inference outcome of model-data misalignment was systematically investigated (see Sections 5.1 to 5.5). The results of these scenarios, including scenario F mentioned above, in terms of convergence, model fit and inferred noise level, have been summarized in Table 6.1[1].

As can be seen from Table 6.1, the results show that under the circumstance that the model and observed data are in near perfect alignment (scenario A and E), and accurate prior information on maximum peak location is passed to the model, then convergence and model fit are very good, and the inferred noise level is nearly equal to the real noise level. However, even in this case, the results are strongly influenced by the choice of algorithm initialisation method, prior maximum peak location information, prior parameter shape, and the number of parallel threads used for the inference process (see Section 5.1).

When varying the baseline shape and the number of peaks present in the generated data (scenario B and C), two distinct situations occur. In the first situation, the probabilistic model has more parameters available than the number strictly needed to

[1]The table values are mapped using the following boundaries:
$\hat{R} \leq 1.1$ (good), $1.1 < \hat{R} < 1.3$ (neutral), $\hat{R} \geq 1.3$ (bad)
$R^2 \geq 0.99$ (good), $0.99 > R^2 > 0.95$ (neutral), $R^2 \leq 0.95$ (bad)
$\sigma_\epsilon \simeq$ 1x real value (good), $1x$ real value $< \sigma_\epsilon <$ 5x real value (neutral), $\sigma_\epsilon \geq$ 5x real value (bad)

explain the data, corresponding to an over-specified situation. In the case of baseline variation (scenario B, see Section 5.2), this leads to a situation where convergence and model fit are good, and the inferred noise level is in agreement with the real value. In the case of peak number variation (scenario C, see Section 5.3), this results in a good model fit and a noise level which is also in agreement with the real value. However, in this case the convergence measures are not good, as multiple solutions exist for the inference of the excess peak locations. In the second situation, the model is lacking in parameters to accurately describe the data, corresponding to an under-specified situation. Baseline variation in this case leads to bad convergence, bad model fit and high noise levels which are much higher than the noise levels in the generated data. In the case of peak number variation, this situation leads to a good convergence, but a poor model fit and high noise levels not comparable with the real value.

For a model with a fixed peak shape factor, where the peak shape factor was varied in the data (scenario D, see Section 5.4), convergence is good for all model-data combinations. However, model fit decreases and noise levels increase, as the discrepancy between the peak shape fixed in the model and the peak shape set in the data grows larger. In the situation where the peak shape factor is added as a random variable to the model (scenario E, see Section 5.5), convergence and model fit are good, and the noise level and peak shape factor generally are in accordance with the real values as set in the data.

In summary, in case the probabilistic model is able to adapt the parameters to fit to the data, as is the case for scenarios A and E, all balanced conditions in scenarios B, C and D, and in the over-specified regions in scenario B and C, then the inferred noise level is generally in agreement with the real noise value set in the data. In case the model is unable to adapt the parameters to fit to the data, as is the case for the under-specified region in scenarios B and C, the major effect seen as a result from an induced misalignment between the probabilistic model and the generated datasets is an increase of the noise level. As the set of possible solutions resulting from the combination of model parameters diverges from the observed data, increasing the noise level minimizes the resulting error between model prediction and data. Furthermore, it is assumed that the same process is responsible for the high noise levels seen from inference on real-world datasets. This implies that the current probabilistic model is still misaligned with the hidden, generative processes behind observed real-world datasets. A list of recommendations for improving the current probabilistic model, thereby decreasing the model-data misalignment is discussed further in Section 6.3.

With the two research questions answered, the main research objective can now also be answered. The main research objective, was formulated as follows:

Can probabilistic programming be used for spectroscopic data analysis?

The underlying assumption behind the main research objective is that spectroscopic data analysis includes the possibility for classification of chemical mixtures, as the main research objective was formulated to investigate the possibility to overcome limitations which originated from CNN based spectroscopic data analysis (see Section 1.2).

Identifying a separate class within a chemical mixture requires the identification of specific features unique to the class. These features are composed of distinct peaks, or combination of peaks, which fingerprint the desired chemical component (see Section 1.1).

The first step in the development of this capability is the ability to capture features from the vibrational spectrum, and this has been the focus of the first research question. However, the noise levels seen after parameter inference on real-world vibrational datasets are currently not in agreement with the noise levels as observed from the spectrum directly, indicating a structural misalignment between the model and data. Although broad spectral features which lie above the inferred noise level can probably be detected and quantified, distinct spectral features which are characteristic of a specific chemical class, are often much smaller in amplitude and have a high probability of falling below the inferred noise level, making them undetectable. Therefore, with the current probabilistic model and the current results the answer is undecided, and no definitive answer can be given yet to the main research objective, as improvements in the probabilistic model could still significantly improve the results on real-world datasets.

In hindsight, the current research approach proved to be too optimistic. By directly applying a probabilistic model to real-world datasets, the complexity of real-world data was underestimated, which lead to unsatisfactory results. Even after switching to generated datasets, which was an attempt to get more control over the data, developing and fine-tuning a probabilistic model and re-applying it to real-world datasets, the results remained indecisive. Answering the main research objective will therefore probably require a series of smaller steps. A list of recommendations for future work is given in Section 6.3.

6.2. LIMITATIONS

The results of the current research mainly rely on spectral data that was generated with the dataset generator (see Section 4.1). In turn, the spectral model underlying the dataset generator was built upon results of earlier research on the Bayesian modelling of vibrational spectra (Frøhling et al., 2016; Moores et al., 2016; Han & Ram, 2019).

When the current research is compared to the earlier research, it differs from it mainly in two ways. First, earlier research focuses on the modelling of Raman spectra, which show less spectral artefacts than the IR/NIR spectra used in this research. Second, the chemical mixtures used in earlier research are less complex, consisting of a few pure chemical components at maximum, which reduces the complexity of the mixture spectrum.

Although the dataset generator generates spectra which qualitatively mimic real-world datasets, during the current research no effort was made to quantify this similarity. As real-world vibrational spectroscopic data tends to be very complex, having many peaks of all sizes, often with non-linear baseline behaviour, having a quantitative measure on how well the dataset generator is able to generate real-world spectra would significantly increase the validity of the results. It therefore remains an open question if the current experimental approach has generated results which can be interpolated to real-world data, or that a further increase in dataset generator complexity is needed first.

6.3. FUTURE WORK

With the conclusions and limitations of the current research described, it is possible to identify potential directions for future work. All of these recommendations support in providing an answer to the main research objective, and only after making progress on each one a more definitive answer to the research objective can be given. Three steps, intended to be followed in a sequential order, can be identified for improvement.

First, as was shown in the limitations (see Section 6.2), the validity of the dataset generator can be increased by comparing generated datasets with real-world datasets, and quantifying how well the two sets compare. This can be done by taking a relatively simple (e.g. a spectrum with a small number of clearly distinguishable peaks) and clean (e.g. a spectrum with simple, linear distortions) real-world spectrum, or part of a spectrum, estimating the hidden parameters by hand or from external sources, and generating datasets with these estimated parameter values. In a next step a more complex real-world spectrum (i.e. more peaks and more complex spectral artefacts) can then be measured, simulated and compared.

Second, as the main research objective is to be able to classify vibrational spectroscopic data (see Section 6.1), the dataset generator should be modified so that it is able to generate spectra of different chemical classes, i.e. generate spectra of chemical mixtures. To be able to infer parameter values from these spectra, the probabilistic model should be modified accordingly. This step will then give an indication of the classification performance on simulated datasets.

Third, after having progressed on both areas, parameter inference on real-world spectroscopic datasets can be performed and re-evaluated. Also here, a gradual, from simple to complex approach seems reasonable, e.g. by starting with low-complexity real-world datasets which have clear characteristic features separating the different classes in the chemical mixture, to more complex datasets, in which the characteristic features are less separated.

6.4. SOFTWARE LICENSE

All software belonging to this research is distributed under the MIT-license[2]. The software is available at: `https://github.com/jnispen/PPSDA`.

[2] https://github.com/jnispen/PPSDA/blob/master/LICENSE.txt

A

SUPPLEMENTAL TABLES AND FIGURES

A.1. SCENARIO A — NOISE VARIATION

Table A.1: **LogNormal model —** Convergence per init method and noise level. Inference repeated eight times per noise level and init method (N = 10, peak shift = 0%, cores = 2).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	4 2 4 5 4 2 1 3	8 9 10 9 9 9 9 8	9 9 9 9 9 10 10 10
2%	5 4 2 6 4 4 5 3	9 9 10 9 10 9 10 10	10 10 10 10 10 10 10 10
3%	1 5 1 2 2 3 4 4	9 8 9 8 8 9 9 8	9 9 9 9 9 9 9 9
	10 11 7 13 10 9 10 10	**26 26 29 26 27 27 28 26**	**28 28 28 28 28 29 29 29**

Table A.2: **LogNormal model —** Convergence per init method and noise level. Inference repeated eight times per noise level and init method (N = 10, peak shift = 2%, cores = 2).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	0 2 3 3 1 1 2 2	10 10 10 9 10 9 8 7	10 10 9 9 10 9 9 9
2%	1 5 2 1 1 2 3 1	9 8 8 8 8 8 8 7	10 10 10 10 10 9 10 10
3%	2 1 1 1 2 1 1 2	6 8 6 8 6 8 9 8	9 9 9 9 9 9 9 9
	3 8 6 5 4 4 6 5	**25 26 24 25 24 25 25 22**	**29 29 28 28 29 27 28 28**

Table A.3: **LogNormal model —** Convergence per init method and noise level. Inference repeated eight times per noise level and init method (N = 10, peak shift = 5%, cores = 2).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	1 2 3 2 2 1 0 2	7 6 7 6 7 7 7 6	9 9 8 8 8 9 9 8
2%	1 2 3 2 2 2 3 3	8 8 8 8 8 8 8 9	7 8 6 8 6 9 8 8
3%	0 1 1 0 1 0 3 2	5 6 7 6 7 5 8 5	7 8 7 5 7 8 8 7
	2 5 6 4 5 3 6 7	**20 20 22 20 22 20 23 20**	**23 25 21 21 21 26 25 23**

Table A.4: **LogNormal model —** Convergence per init method and noise level. Inference repeated eight times per noise level and init method (N = 10, peak shift = 10%, cores = 2).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	0 1 0 1 0 0 1 2	1 4 2 2 2 3 3 1	3 4 2 0 1 5 4 4
2%	3 1 2 2 0 4 1 1	6 4 2 5 5 6 4 4	3 6 3 4 3 3 5 1
3%	1 0 2 1 4 1 1 4	2 3 4 2 0 1 3 2	4 3 2 2 6 4 4 4
	4 2 4 4 4 5 3 7	**9 8 8 9 7 10 10 7**	**10 13 7 6 10 12 13 9**

Table A.5: **LogNormal model —** Convergence per init method and noise level. Inference repeated eight times per noise level and init method (N = 10, no peak location information, cores = 2).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	1 0 0 2 0 0 0 0	1 0 2 2 1 1 1 1	0 1 0 0 0 1 0 2
2%	2 0 0 2 0 1 2 0	1 1 2 1 1 2 3 0	2 2 0 3 2 1 0 1
3%	0 0 0 0 0 0 1 1	2 0 0 3 2 1 0 2	2 0 0 1 1 0 1 0
	3 0 0 4 0 1 3 1	**4 1 4 6 4 4 4 3**	**4 3 0 4 3 2 1 3**

Table A.6: **Normal model —** Convergence per init method and noise level. Inference repeated eight times per noise level and init method (N = 10, peak shift = 0%, cores = 2).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	5 3 3 3 2 2 7 3	6 7 6 6 6 7 7 6	6 6 6 5 5 5 5 6
2%	3 6 5 3 3 4 5 4	4 5 4 2 3 4 4 3	6 5 6 4 6 5 4 5
3%	3 4 4 3 2 4 4 4	4 4 4 4 5 4 4 4	3 4 4 3 3 4 5 4
	11 13 12 9 7 10 16 11	**14 16 14 12 14 15 15 13**	**15 15 16 12 14 14 14 15**

Table A.7: **Normal model —** Convergence per init method and noise level. Inference repeated eight times per noise level and init method (N = 10, no peak location information, cores = 2).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	1 2 1 1 1 3 0 0	0 2 1 0 2 2 2 1	2 3 0 1 1 3 1 2
2%	0 2 1 1 0 1 1 0	1 2 1 2 2 2 2 2	0 0 0 0 0 0 0 0
3%	1 1 0 0 0 0 0 1	1 0 0 1 1 1 0 1	1 2 3 0 0 0 1 0
	2 5 2 2 1 4 1 1	**2 4 2 3 5 5 4 4**	**3 5 3 1 1 3 2 2**

Table A.8: **LogNormal model —** Convergence per init method and noise level. Inference repeated four times per noise level and init method (N = 10, peak shift = 0%, cores = 6).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	0 0 0 0	8 9 8 9	9 9 9 9
2%	2 0 2 1	9 10 9 9	10 10 10 10
3%	0 0 0 1	7 8 8 8	9 9 9 9
	2 0 2 2	**24 27 25 26**	**28 28 28 28**

Table A.9: **LogNormal model —** Convergence per init method and noise level. Inference repeated four times per noise level and init method (N = 10, peak shift = 0%, no peak information, cores = 6).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	0 0 0 0	0 0 0 0	0 0 0 0
2%	0 0 0 0	0 0 0 0	0 0 0 0
3%	0 0 0 0	0 0 0 0	0 0 0 0
	0 0 0 0	**0 0 0 0**	**0 0 0 0**

Table A.10: **Normal model —** Convergence per init method and noise level. Inference repeated four times per noise level and init method (N = 10, peak shift = 0%, cores = 6).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	1 0 2 2	6 5 5 5	6 4 5 3
2%	0 1 0 0	3 3 4 3	4 5 5 4
3%	2 1 1 1	4 4 4 4	3 3 4 3
	3 2 3 3	**13 12 13 12**	**13 12 14 10**

Table A.11: **Normal model —** Convergence per init method and noise level. Inference repeated four times per noise level and init method (N = 10, peak shift = 0%, no peak information, cores = 6).

	init method		
Noise	**jitter+adapt_diag**	**advi+adapt_diag**	**adapt_diag**
1%	1 1 0 0	1 1 1 1	1 1 1 1
2%	0 0 0 0	1 1 1 0	0 0 0 0
3%	0 0 0 0	1 1 0 0	0 0 0 0
	1 1 0 0	**3 3 2 1**	**1 1 1 1**

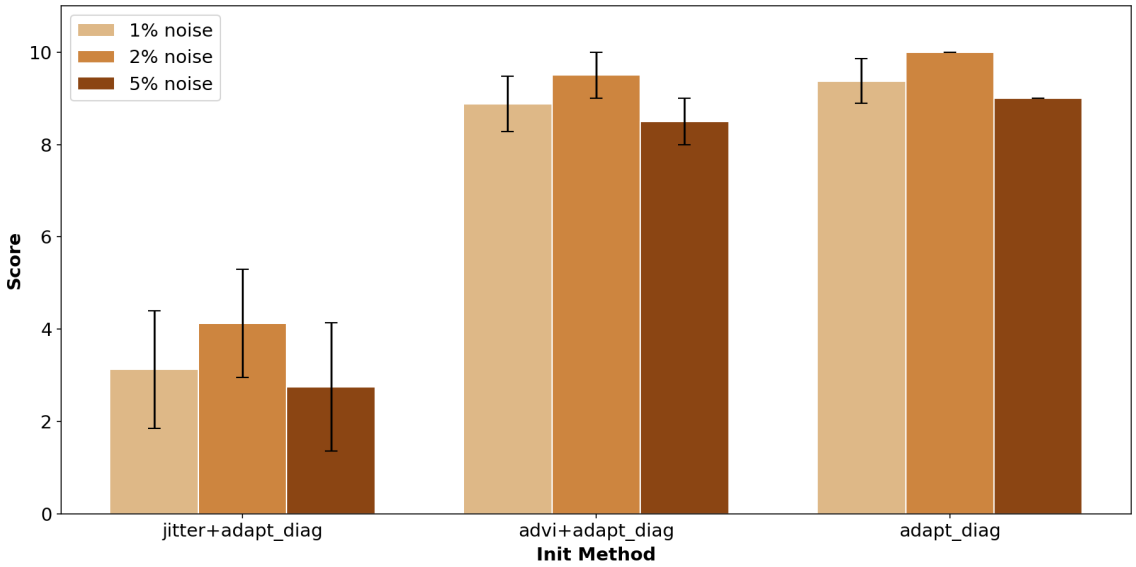


Figure A.1: Sampling convergence for LogNormal model per noise level and init method (N = 10, peak shift = 0%).

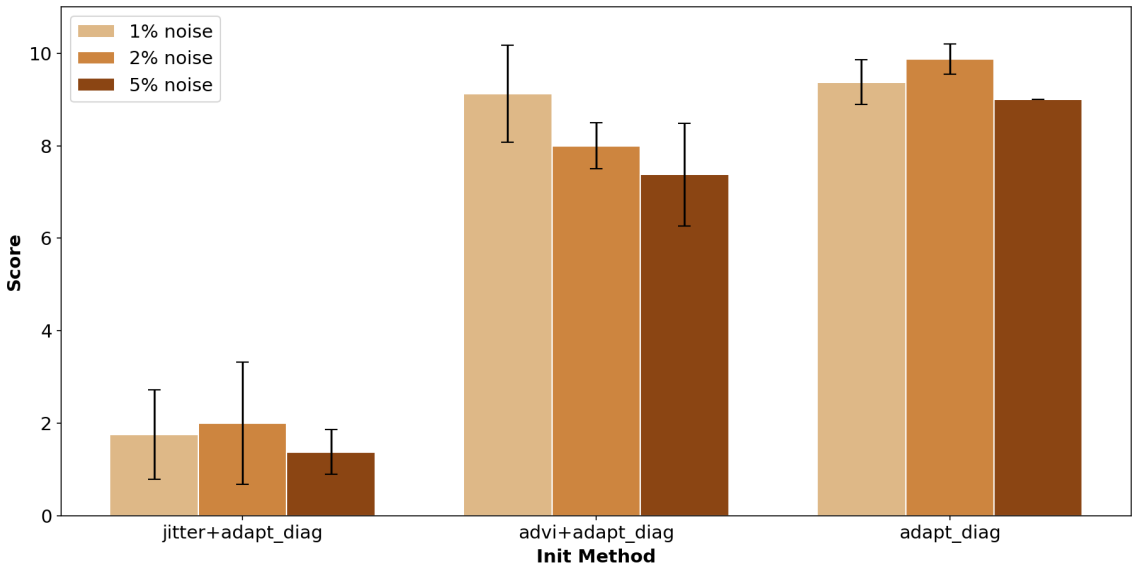


Figure A.2: Sampling convergence for LogNormal model per noise level and init method (N = 10, peak shift = 2%).

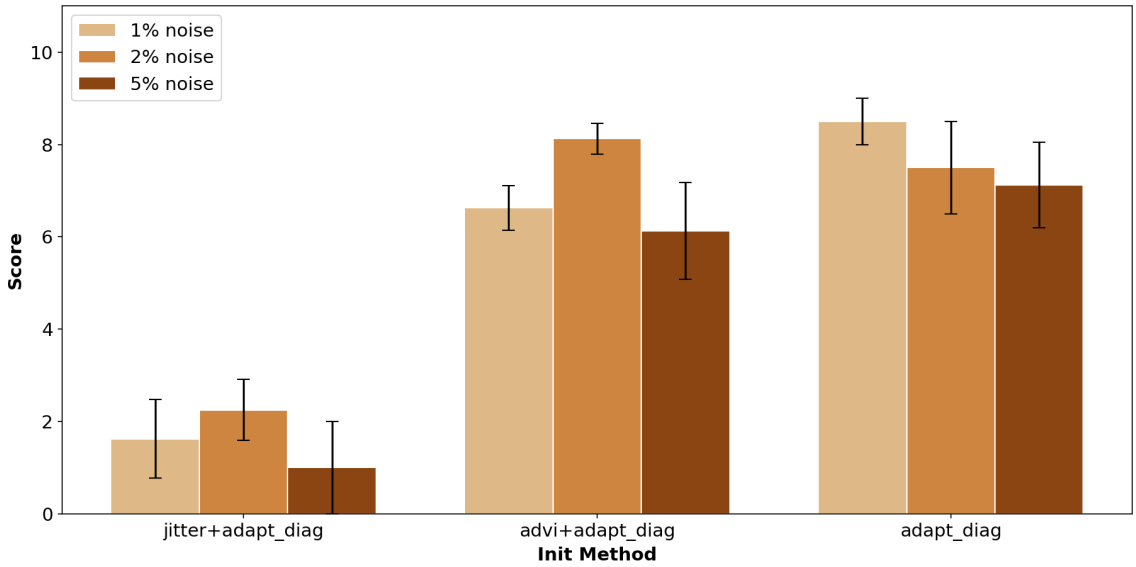


Figure A.3: Sampling convergence for LogNormal model per noise level and init method (N = 10, peak shift = 5%).

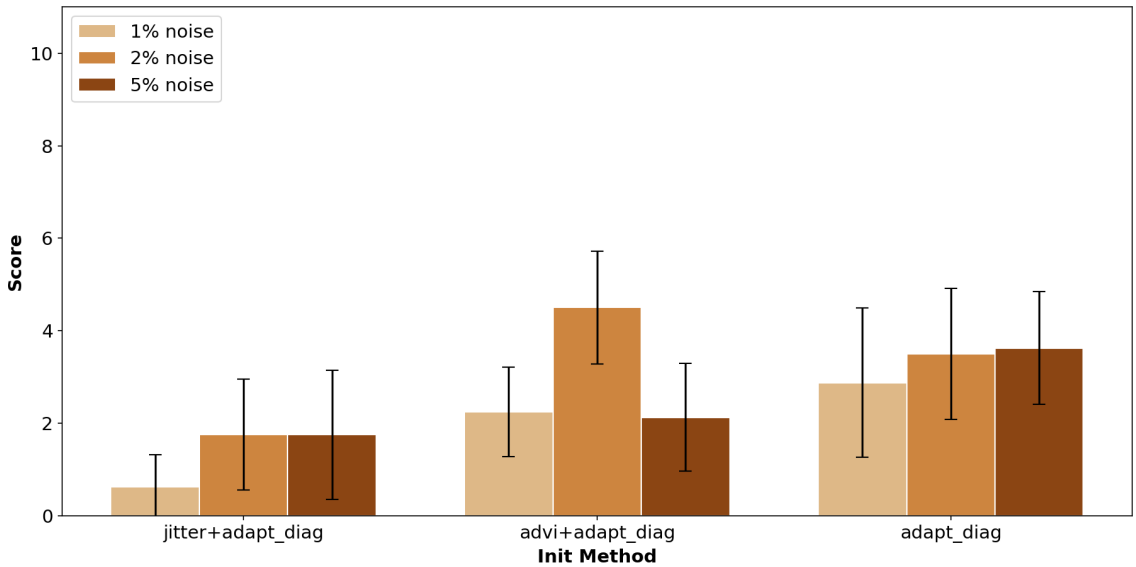


Figure A.4: Sampling convergence for LogNormal model per noise level and init method (N = 10, peak shift = 10%).

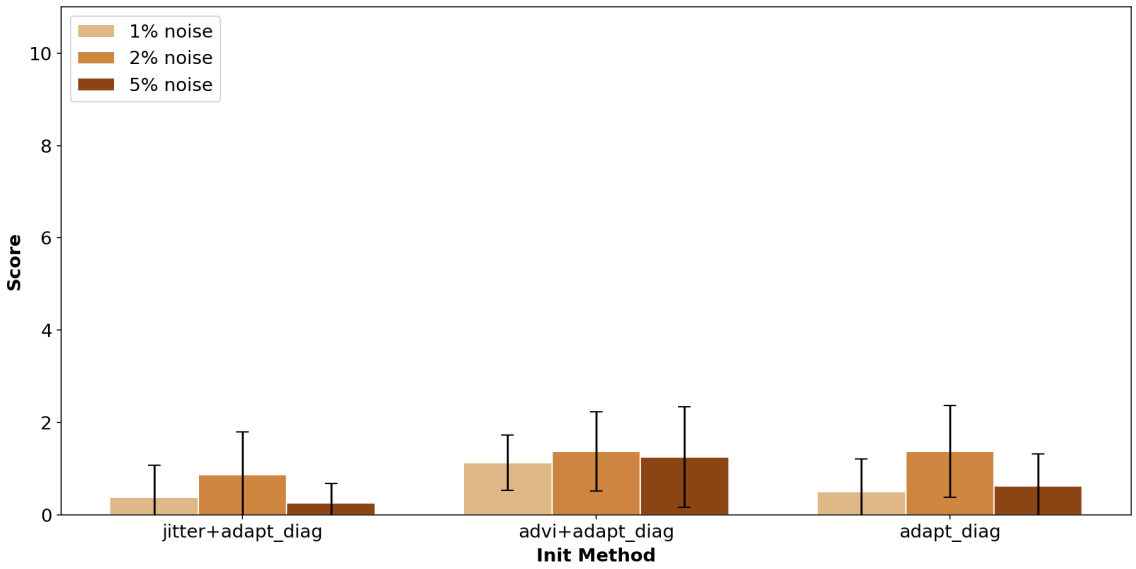


Figure A.5: Sampling convergence for LogNormal model per noise level and init method (N = 10, no peak location information).

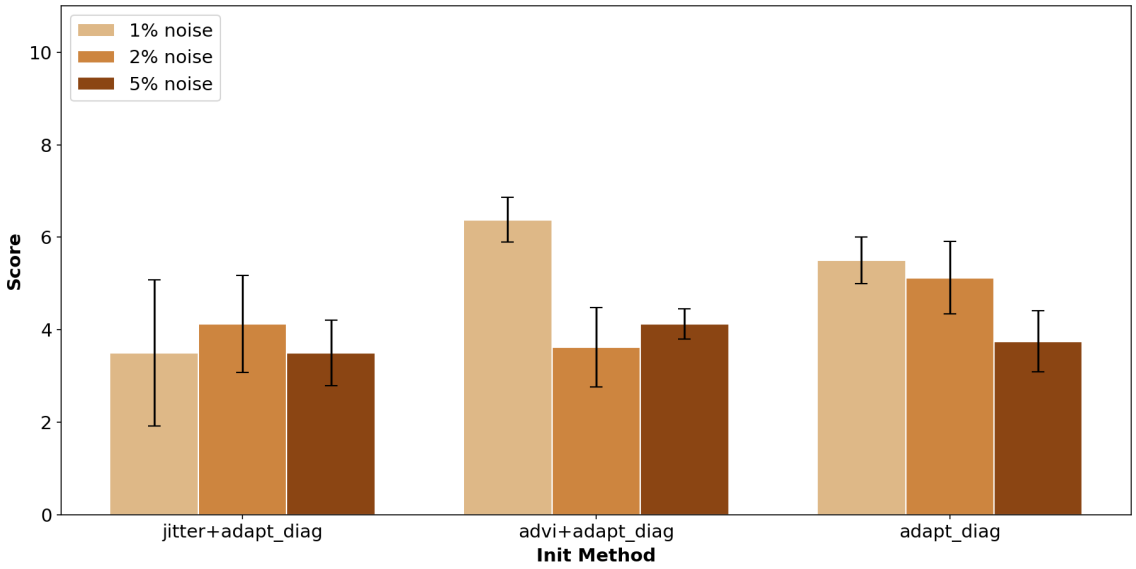


Figure A.6: Sampling convergence for Normal model per noise level and init method (N = 10, peak shift = 0%).

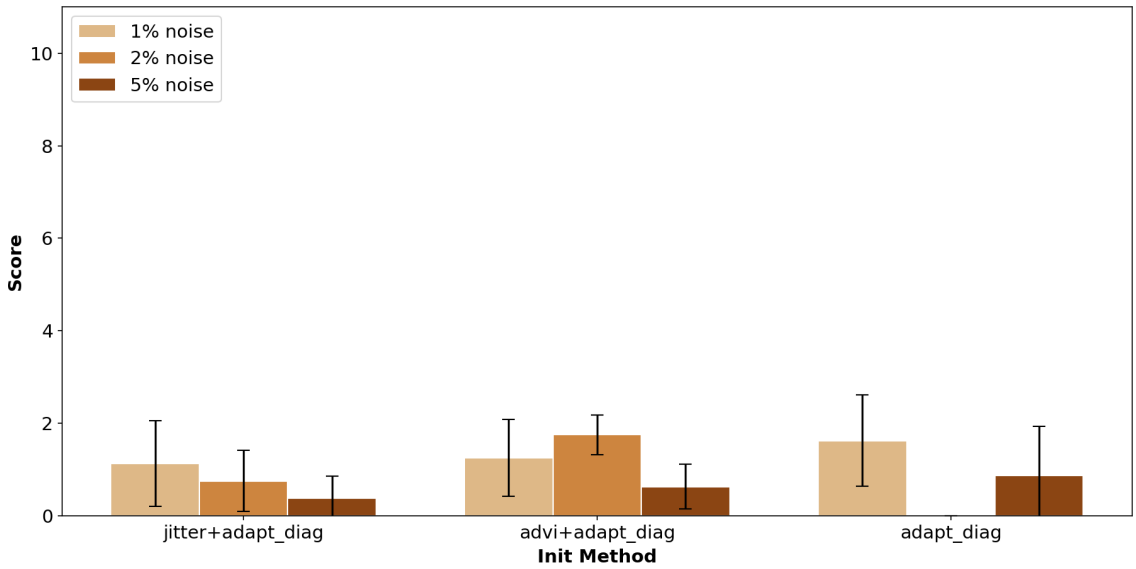
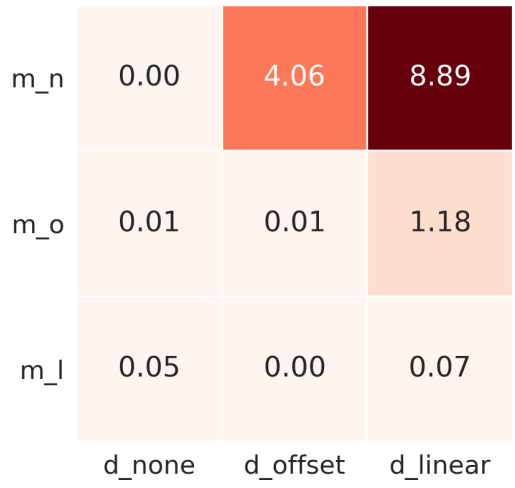
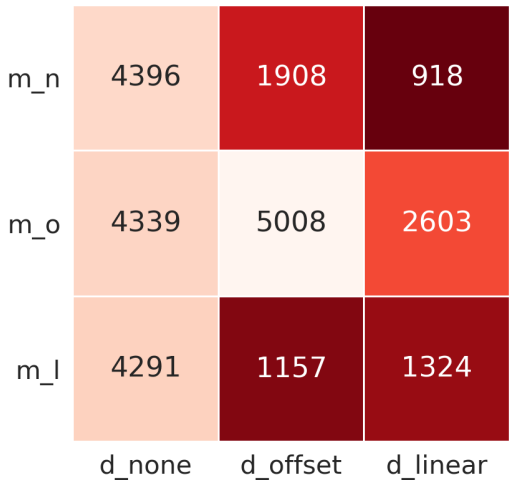


Figure A.7: Sampling convergence for Normal model per noise level and init method (N = 10, no peak location information).

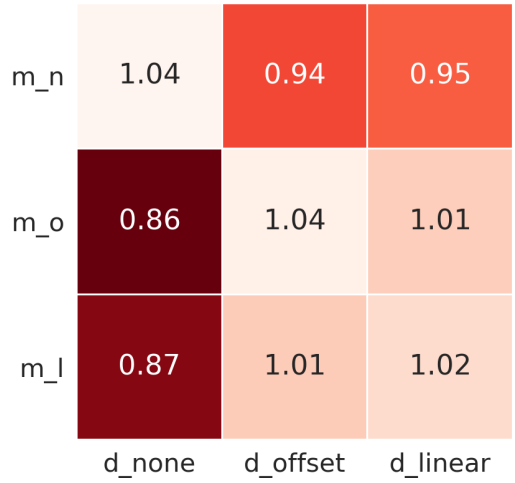
A.2. SCENARIO B — BASELINE VARIATION



(a) MCSE



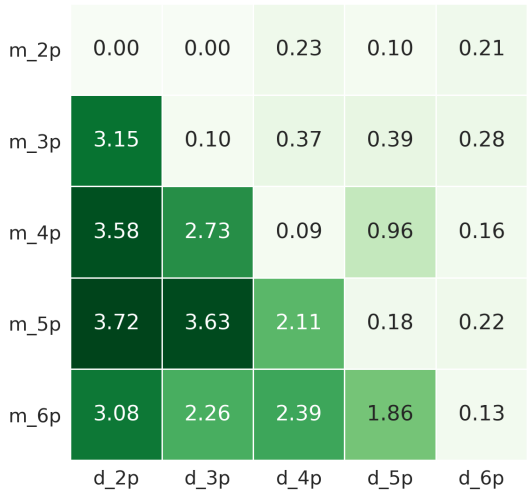
(b) ESS (> 1000 is good)



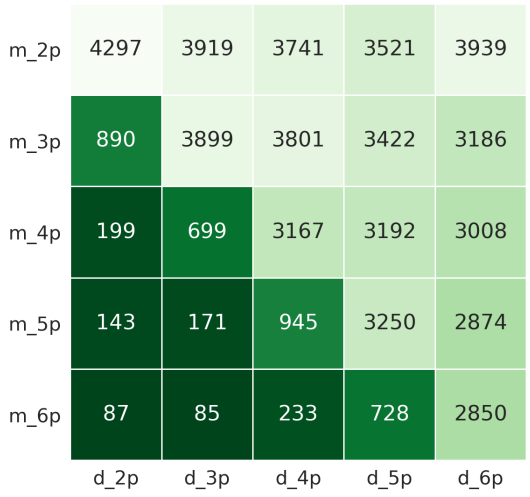
(c) BFMI (≈ 1 is good)

Figure A.8: Evaluation measures for scenario B (Baseline variation). The subfigures (a)–(c) show different evaluation measures for nine possible model-data combinations. Unless marked, lower intensity color indicates a better performance of the measure.

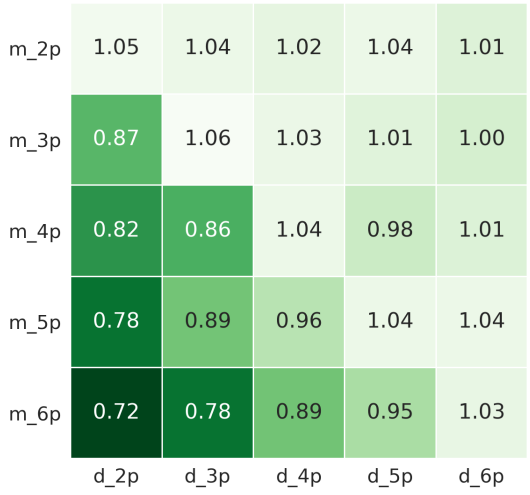
A.3. SCENARIO C — PEAK NUMBER VARIATION



(a) MCSE



(b) ESS (> 1000 is good)



(c) BFMI (≈ 1 is good)

Figure A.9: Evaluation measures for scenario C (Peak number variation). The subfigures (a)–(c) show different evaluation measures for 25 possible model-data combinations. Unless marked, lower intensity color indicates a better performance of the measure.

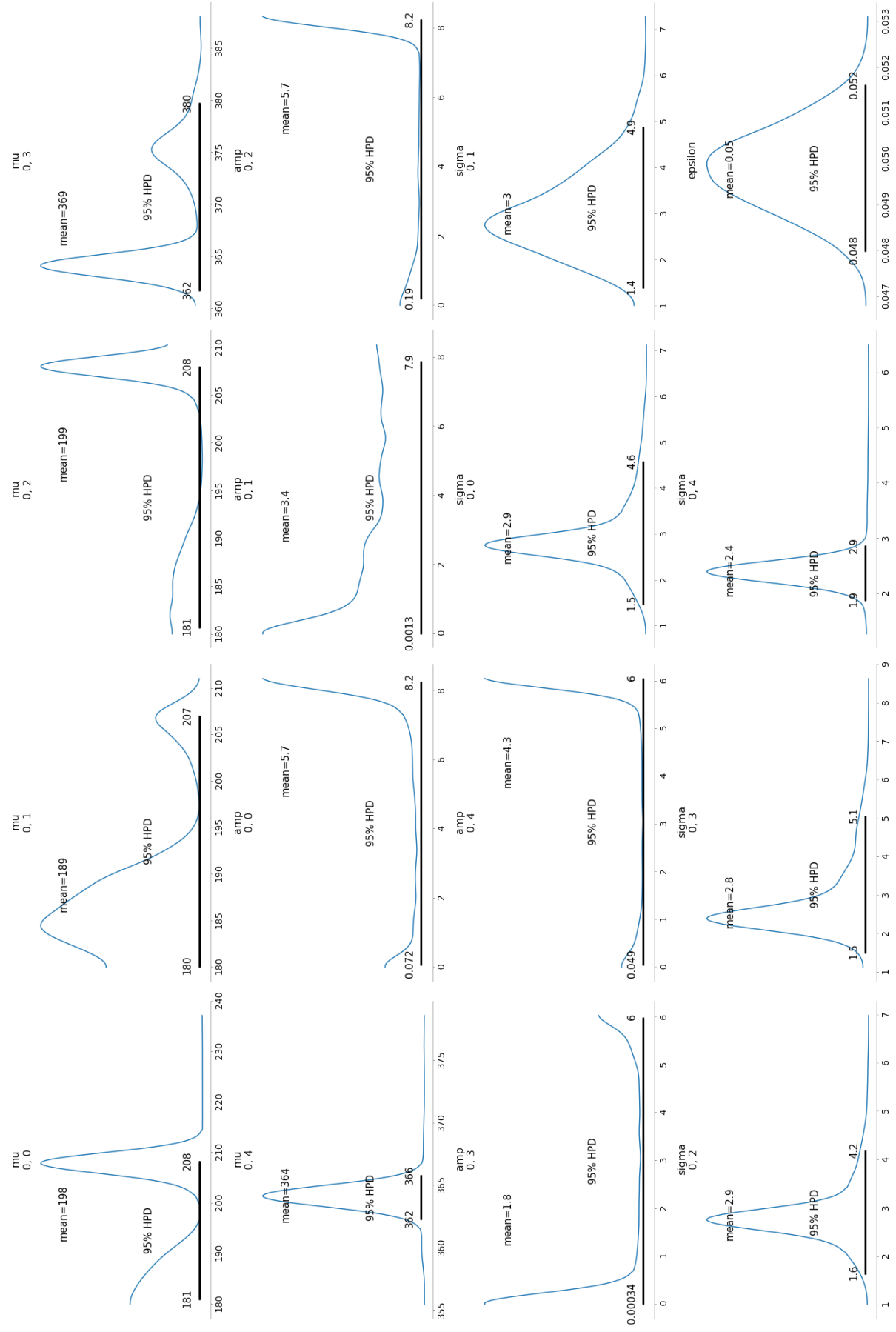


Figure A.10: Posterior distributions of the model parameters from Figure 5.5 (a).

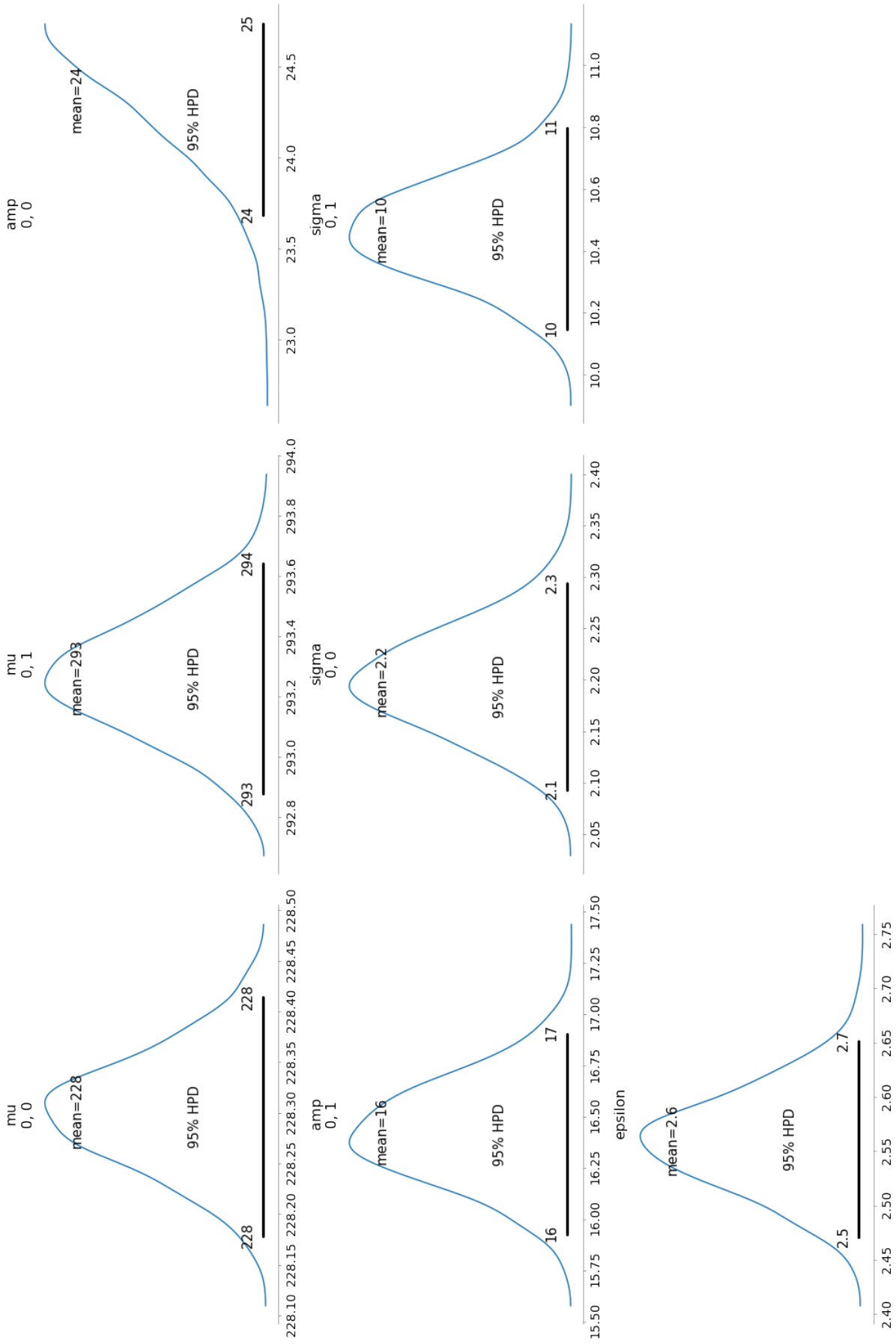
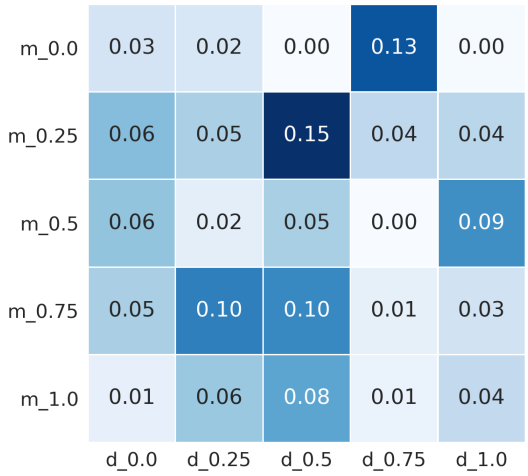
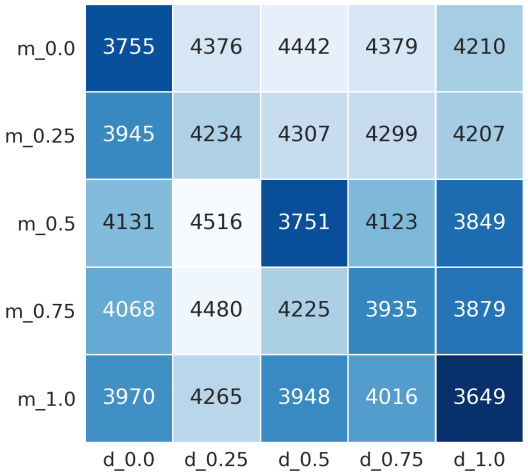


Figure A.11: Posterior distributions of the model parameters from Figure 5.5 (b).

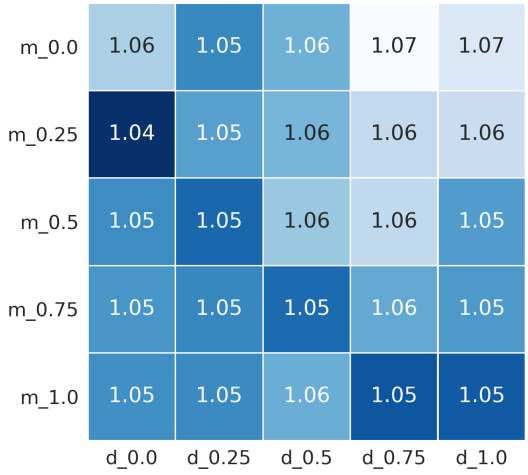
A.4. SCENARIO D — PEAK SHAPE VARIATION I



(a) MCSE



(b) ESS (> 1000 is good)



(c) BFMI (≈ 1 is good)

Figure A.12: Evaluation measures for scenario D (Peak shape variation I). The subfigures (a)–(c) show different evaluation measures for 25 possible model-data combinations. Unless marked, lower intensity color indicates a better performance of the measure.

REFERENCES

Acquarelli, J., van Laarhoven, T., Gerretzen, J., Tran, T., Buydens, L., & Marchiori, E. (2017). Convolutional neural networks for vibrational spectroscopic data analysis. *Analytica chimica acta*, *954*, 22–31.

Acquarelli, J., van Laarhoven, T., Jansen, J., Buydens, L., & Marchiori, E. (2019). *Deep learning for spectroscopic data analysis: a critical assessment.* (preprint)

Betancourt, M. (2016). Diagnosing suboptimal cotangent disintegrations in hamiltonian monte carlo. *arXiv preprint arXiv:1604.00695*.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Bjerrum, E. J., Glahder, M., & Skov, T. (2017). Data augmentation of spectral data for convolutional neural network (cnn) based deep chemometrics. *arXiv preprint arXiv:1710.01927*.

Blei, D. M. (2014). Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, *1*, 203–232.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, *112*(518), 859–877.

Box, G. E. (1976). Science and statistics. *Journal of the American Statistical Association*, *71*(356), 791–799.

Bradley, M. (2007). Curve fitting in raman and ir spectroscopy: basic theory of line shapes and applications. *Thermo Fisher Scientific, Madison, USA, Application Note, 50733*.

Davidson-Pilon, C. (2015). *Bayesian methods for hackers: probabilistic programming and bayesian inference*. Addison-Wesley Professional.

Doucet, A., De Freitas, N., & Gordon, N. (2001). An introduction to sequential monte carlo methods. In *Sequential monte carlo methods in practice* (pp. 3–14). Springer.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, *195*(2), 216–222.

Dyrby, M., Engelsen, S. B., Nørgaard, L., Bruhn, M., & Lundsberg-Nielsen, L. (2002). Chemometric quantitation of the active substance (containing c≡ n) in a pharmaceutical tablet using near-infrared (nir) transmittance and nir ft-raman spectra. *Applied Spectroscopy*, *56*(5), 579–585.

Engel, J., Gerretzen, J., Szymańska, E., Jansen, J. J., Downey, G., Blanchet, L., & Buydens, L. M. (2013). Breaking with trends in pre-processing? *TrAC Trends in Analytical Chemistry*, *50*, 96–106.

Flegal, J. M., Haran, M., & Jones, G. L. (2008). Markov chain monte carlo: Can we trust the third significant figure? *Statistical Science*, 250–260.

Frøhling, K. B., Alstrøm, T. S., Bache, M., Schmidt, M. S., Schmidt, M. N., Larsen, J., ... Boisen, A. (2016). Surface-enhanced raman spectroscopic study of dna and 6-mercapto-1-hexanol interactions using large area mapping. *Vibrational Spectroscopy*, *86*, 331–336.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis*. CRC press.

Gelman, A., Goodrich, B., Gabry, J., & Vehtari, A. (2019). R-squared for bayesian regression models. *The American Statistician*, *73*(3), 307–309.

Gelman, A., Hwang, J., & Vehtari, A. (2014). Understanding predictive information criteria for bayesian models. *Statistics and computing*, *24*(6), 997–1016.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, *521*(7553), 452.

Han, N., & Ram, R. J. (2019). Bayesian modeling and computation for analyte quantification in complex mixtures using raman spectroscopy. *Computational Statistics & Data Analysis*, 106846.

Hastings, W. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, *57*, 97–109.

Hoffman, M. D., & Gelman, A. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, *15*(1), 1593–1623.

Kruschke, J. (2014). *Doing bayesian data analysis: A tutorial with r, jags, and stan*. Academic Press.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., & Blei, D. M. (2017). Automatic differentiation variational inference. *The Journal of Machine Learning Research*, *18*(1), 430–474.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

Lerma-García, M., Ramis-Ramos, G., Herrero-Martínez, J., & Simó-Alfonso, E. (2010). Authentication of extra virgin olive oils by fourier-transform infrared spectroscopy. *Food Chemistry*, *118*(1), 78–83.

Marini, F. (2009). Artificial neural networks in foodstuff analyses: Trends and perspectives a review. *Analytica Chimica Acta*, *635*(2), 121–131.

Martin, O. (2018). *Bayesian analysis with python: Introduction to statistical modeling and probabilistic programming using pymc3 and arviz*. Packt Publishing Ltd.

Massart, D. L., Vandeginste, B. G., Buydens, L., De Jong, S., Lewi, P. J., Smeyers-Verbeke, J., & Mann, C. K. (1998). Handbook of chemometrics and qualimetrics: Part a. *Applied Spectroscopy*, *52*, 302A.

Meinshausen, N., & Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*(4), 417–473.

Moores, M., Gracie, K., Carson, J., Faulds, K., Graham, D., & Girolami, M. (2016). Bayesian modelling and quantification of raman spectroscopy. *arXiv preprint arXiv:1604.07299*.

Pfeffer, A. (2016). *Practical probabilistic programming*. Manning Publications Co.

Rinnan, Å., Van Den Berg, F., & Engelsen, S. B. (2009). Review of the most common preprocessing techniques for near-infrared spectra. *TrAC Trends in Analytical Chemistry*, *28*(10), 1201–1222.

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in python using pymc3. *PeerJ Computer Science*, *2*, e55.

Stuart, B. (2004). Infrared spectroscopy: fundamentals and applications. *John Wiley and Sons, Ltd., West Sussex, England. DOI*, *10*, 0470854278.

van de Meent, J.-W., Paige, B., Yang, H., & Wood, F. (2018). An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*.

Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical bayesian model evaluation using leave-

one-out cross-validation and waic. *Statistics and computing*, *27*(5), 1413–1432.

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., & Bürkner, P.-C. (2019). Rank-normalization, folding, and localization: An improved ^ r for assessing convergence of mcmc. *arXiv preprint arXiv:1903.08008*, 18.

Watanabe, S. (2010). Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research*, *11*(Dec), 3571–3594.

Wertheim, G., Butler, M., West, K., & Buchanan, D. (1974). Determination of the gaussian and lorentzian content of experimental line shapes. *Review of Scientific Instruments*, *45*(11), 1369–1371.

Yang, J., Xu, J., Zhang, X., Wu, C., Lin, T., & Ying, Y. (2019). Deep learning for vibrational spectral analysis: Recent progress and a practical guide. *Analytica Chimica Acta*.

Yun, Y.-H., Li, H.-D., Deng, B.-C., & Cao, D.-S. (2019). An overview of variable selection methods in multivariate analysis of near-infrared spectra. *TrAC Trends in Analytical Chemistry*.

ACRONYMS

ADVI	Automatic Differentiation Variational Inference
ANN	Artificial Neural Network
BFMI	Bayesian Fraction of Missing Information
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
ELBO	Evidence Lower Bound
ESS	Effective Sample Size
FTIR	Fourier Transform Infrared
FWHM	Full Width at Half Maximum
HMC	Hamiltonian Monte Carlo
HPD	Highest Posterior Density
IR	Infrared
KL	Kullback-Leibler divergence
kNN	k-Nearest Neighbors
LDA	Linear Discriminant Analysis
LogReg	Logistic Regression
LOO	Leave-One-Out cross-validation
MCMC	Markov chain Monte Carlo
MCSE	Monte Carlo Standard Error
MH	Metropolis-Hastings
MVC	Multivariate Calibration
NIR	Near Infrared
NUTS	No-U-Turn Sampler
PCA	Principal Component Analysis
PCR	Principle Component Regression
PGM	Probabilistic Graphical Model
PLS	Partial Least Squares Regression
PPC	Posterior Predictive Checks
PPL	Probabilistic Programming Language
RJMCMC	Reversible-jump Markov chain Monte Carlo
RR	Ridge Regression
SERS	Surface Enhanced Raman Spectroscopy (or Scattering)
SMC	Sequential Monte Carlo
SVM	Support Vector Machine
VI	Variational Inference
WAIC	Widely Applicable Information Criterion